System Aware Cybersecurity
A Multi-Sentinel Scheme To Protect a Weapons Research Lab

Technical Report SERC-2015-TR-110

December 7, 2015

PI: Dr. Barry Horowitz, University of Virginia

Research Team

Co-PI: Dr. Peter Beling, University of Virginia

Co-PI: Dr. Marty Humphrey, University of Virginia

Co-PI: Lt Col Chris Gay, UVa Graduate Student

Contract No. HQ0034-13-D-0004

Task Order: 0036, RT 136

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171 (Task Order 041, RT 141). SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

Table of Contents

Introduction

The Goals of UVa's Systems Aware *Cybersecurity for Physical Systems* Efforts:

We seek an added layer of security to protect the most critical physical system functions

We monitor for illogical system behavior and, upon detection, reconfigure to compensate

We build on cybersecurity, fault tolerant and automatic control technologies

We seek economy through monitoring and reconfiguring a highly secure Sentinel—typically with many more security features than the system being protected can economically employ

We address not only network-based attacks, but also insider and supply chain attacks

We implement reusable design patterns to enable more economical solution development

We use risk-based support tools involving perspectives of both defenders and attackers

(Source: Dr. Horowitz Presentation at DoD (Sept 18th, 2015))

For the Calendar year 2015, the UVa work (as defined by Part I, Section 4 of the RT-136/SERC Research Topic document) fell into four (4) main areas of research: 1) the development of multi-sentinel architectures, 2) the consideration of various Human Factors issues, 3) a demonstration of how Cloud-Based Sentinels (along with the resultant integrity monitoring) could be performed, and 4) an assessment of how Advanced System Modeling and Attack Tree Tools can be integrated into a cyber security assessment workflow. An additional RADAR topic was added later (Appendix 12)

Introduction/Abstract of the Multi-Sentinel Architecture Effort

In this initial section, this document (and the project it reports on) presents a scenario and a model where multiple, overlapping, and redundant rings of defenses protect a sensitive government/military installation such as a weapons research laboratory or nuclear reactor facility; (referred to in this document as ''the facility"). These rings of protection seek to utilize the standard '*system aware'* techniques of hardware/software diversity, configuration hopping, data consistency checking, and tactical forensics to offer confidence that the surveillance data and authentication responses are accurate, reliable, and untampered with.

These protective "rings" include UAV full motion video (FMV) surveillance and ground-based RADAR to detect approaching threats from the air and also ground vehicles from a distance, and a wide variety of physical barriers (fences, walls, locked doors), video surveillance (camera/video feeds), electronic sensors (seismic, acoustic, and motion sensors), as well as RFID/proximity badges for human guards that reinforce the electronic and technical methods of intruder detection, access control, and authentication. The RADAR data is an additional input source in this scenario, and greatly enhances the range that can be covered.

The outer rings of protection are made up of layers 0, 0.5, and 1 (described below as the UAV, the RADAR, and perimeter fence) that seek to either identify approaching entities or to limit access to the facility. The UAV not only scans the ground for ground-based threats and approaching targets, but

also frequently scans the outside of the building to offer the video monitors an opportunity to insure that everything appears normal on the outside of the building, the fence perimeter is intact, and that the RADAR is scanning properly. The RADAR is calibrated to detect approaching aerial and ground vehicle traffic, and the UAV is programmed to fly low enough to be periodically scanned by the RADAR as a check to see if the RADAR is detecting real targets, and that the UAV is still circling overhead in the location expected. The fence line (layer 1) can be reinforced with an underground fiberoptic cable that circles the fence line that detects seismic activity of a human generated origin (foot traffic, approaching vehicles, underground tunneling efforts, etc.) and the fence itself is equipment with motion detectors that would detect climbing activity and/or fence cutting actions. Further details are contained in the layer descriptions on the following pages.

Protective layers 2, 3, and 4 cover the spaces just inside the perimeter fence to the outside walls and doors of the building itself. These protective layers include FMV video cameras, seismic/acoustic/motion sensors, the physical wall itself, and the two entry doors that have biometrically-based authentication/authorization means that are backed up by a human guard at each entry point, each guard employed by a different contractor company or Federal Agency. As the human guards walk their rounds around the interior of the building and around the perimeter, an RFID tag in their badge provides their location information so that the seismic, acoustic, and motion detectors do not flag this activity as originating from a possible intruder. The human guards, when reporting each day for their duty shift, use their RFID-capable badges to scan in at the outer fence line and at the exterior door. Furthermore, additional biometric-based authentication is required to gain complete access to the building's interior.

Although not shown in this document, other layers of interior doors (providing access to the laboratories themselves) should require additional biometric-based authentication (using a different biometric means) and additional human guard scrutiny in order to gain access. In a full production environment, the biometric data, the RFID tag in the guard's badge, and human recognition must all 'match' in order for full access to be given. The 'simplified' scenario B (protecting an internal space) that is discussed in detail in this document can be easily demonstrated with two sets of sensors and associated hardware. Even if not part of the defenses thus far deployed as part of this project, all of these protective technologies have been demonstrated in other venues.

Layer 0: Optional RC/SW controlled UAV for patrolling of airspace and tipping/queuing of approaching air (or ground) vehicles or approaching people. The UAV also carries a SIGINT payload to pickup and identify cell phones, aircraft remote control signals, IFF, scanning radars, etc.

Layer 0.5: RC/SW controlled RADAR to warn of airborne threats, which has its own compass overlay to report the direction of RADAR transmission. Distant RADAR detection sensors report on RADAR parameters and signal strength. These RADAR detectors can also have a built in ADF broadcast signal to aid in UAV navigation independent of GPS signals (or other commonly used navigation signals like VOR/DME, TACAN, etc, and (possibly) some SIGINT detection capability, as well).

Layer 1: An 8-10 foot fence with sensing elements that detect climbers, chain link motion (except for wind), diggers as well as an underground fiberoptic cable (with frequency hopping features) that detect seismic anomalies of human origin

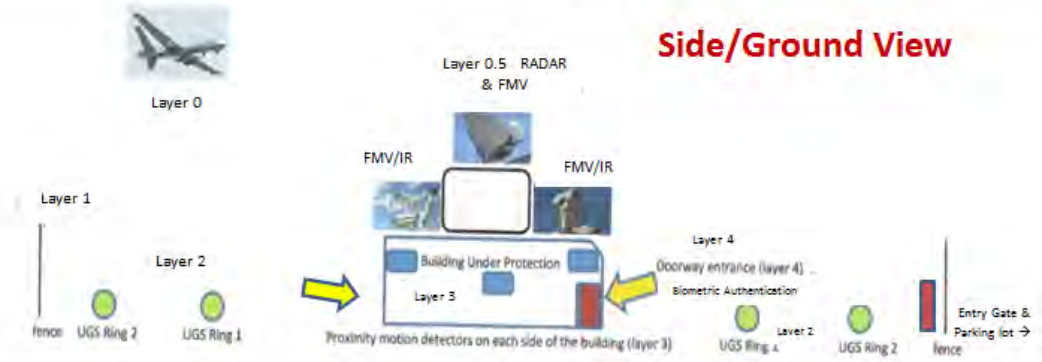
Layer 2: A mixture of acoustic and seismic unattended ground sensors (UGS, both regular sensors and 'sentinel' sensors) for tipping and queuing of infrared sensors or FMV with floodlights (at night) that detect motion (or sound) in sectors that initiate FMV or infrared sensing. Cameras on the fence perimeter supplement other video feeds.

Layer 3: An additional mixture of acoustic and motion sensors on the sides of the building for tipping and queuing of the same FMV/infrared sensors as in layer 2

Layer 4: Biometric-based authentication on doorway entrances, coupled with RFID/proximity badges that are location-sensed at the entrances and at key points inside. The RFID card also doubles as a badge for verification by a human guard/sentry at the two entry points.

The UAV occasionally trains its FMV or infrared camera(s) on the building itself to insure nothing looks amiss there and that the RADAR operation appears to be nominal.

See page 3: These RADAR detectors also have a built in ADF broadcast signals to aid in UAV navigation independent of GPS signals and (possibly) some SIGINT detection capability. The RADAR will occasionally paint the UAV (on low altitude passes) to aid in the system-aware data integrity verification.



Layer 0 Details: The UAV regularly reports GPS location based on GPS signals, but also calculates position based on ADF (Automatic Direction Finding) signals in the area (from local commercial AM/FM radio transmitters), both from the ADF transmitters co-located with the RADAR signal detectors. A software sentinel compares the GPS location with the ADF-calculated position based on rho/theta calculations as a sanity check. During low altitude passes, the UAV scans the building exterior with its FMV or infrared sensors, seeking anomalies in RADAR operation or fence line integrity.

Layer 0.5 details: The sentinel software regularly compares the RADAR dish directional orientation (using a compass installed on the RADAR mount) with RADAR returns from the RADAR detectors on the valley floor as a sanity check. Frequency, pulse width and other metadata are utilized to verify proper operational mode(s). The distant outer ring of RADAR detectors provide feedback on detected RADAR sweeps from the RADAR on top of the building; ADF beacons & possible SIGINT detectors also provide important features

Layer 1 details: Four (4) 'stick shakers' routinely but randomly shake the fence mesh to insure that this motion is accurately sensed and recorded at both the fence motion sensors and via the submerged fiberoptic cable (that senses changes in the light pulses flowing through the cables based on the seismic activity). FMV cameras are mounted at the fence corners & have a large scan range.

Layer 2 details: Two equivalent arrays of acoustic/seismic/SIGINT sensors report acoustic/seismic/SIGINT detected signals. Guards occasionally walk through this array to test its response and to visually inspect the fence line at the start of each eight (8) hour shift and one other

additional time (at random) during each shift. A sensed activity triggers the local FMV or infrared camera, and the guard's facial image is verified. This document focuses on the layer 2 defenses.

Layer 3 details: Same as layer 2, except these sensors are motion detectors on the building that tip/queue FMV/infrared cameras in an 'alert' manner to highlight the impacted video feed

Layer 4: Biometric authentication for personnel entry, authentication of biometrics compared to RFID proximity badge and visual badge inspection by electronic badge reader and/or human sentry. Biometric authentication is performed at the fence line perimeter and at the front entrance to the facility; ultra-sensitive internal labs have additional biometrically-based authentication methods.

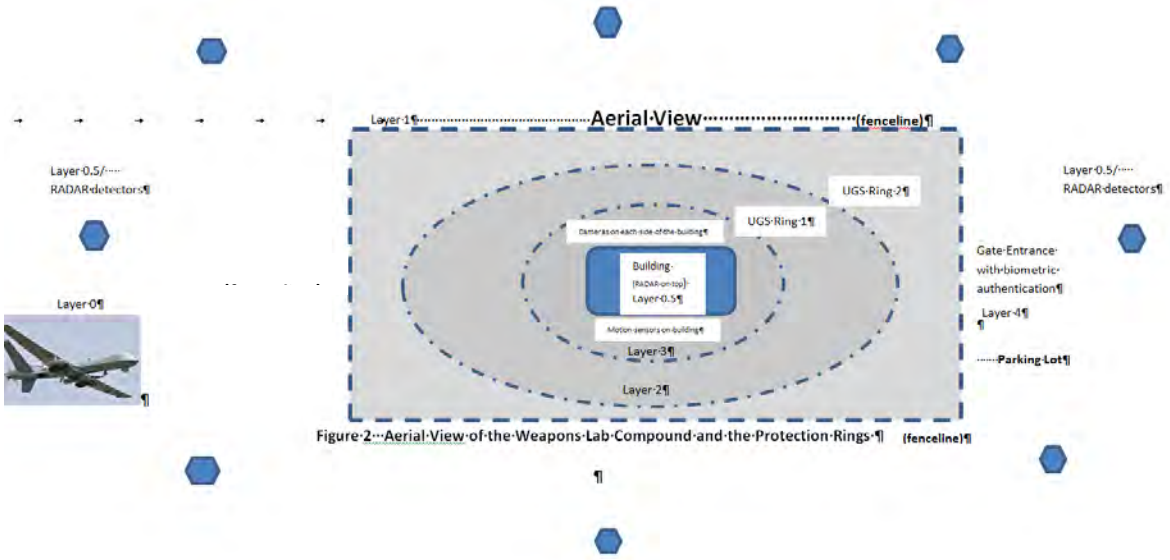
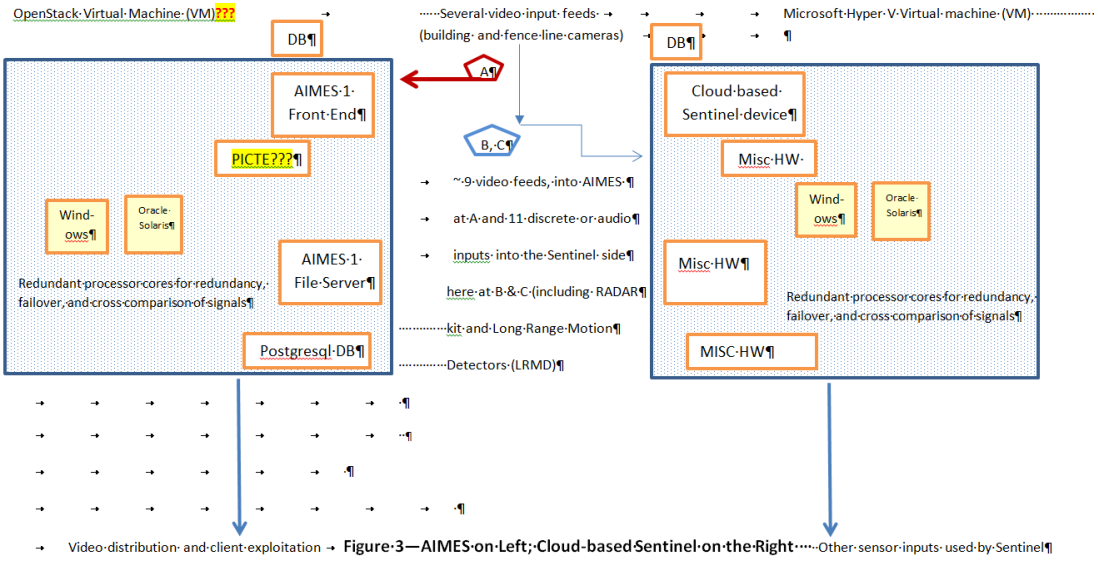


Figure 2 Aerial View of the Weapons Lab Compound and the Protection Rings (fenceline)

The four scanning cameras at the four corners of the fence line randomly scan the perimeter, but if a 'fence shaker' event is detected, the two cameras on these corners focus down the fence line to try and identify the source/driver of the fence's motion.

Figure 3, below, shows the AIMES subsystem and its cloud-based SENTINEL companion subsystem that will be described in greater detail after a discussion of the use cases, general Base Defense Subsystem, etc. But the overall purpose of the Sentinel shown in figure 3 (and ALL Sentinels) is that of monitoring the various subsystems for proper performance and to detect deviations from standard operation, either due to component failure or from external attack. The use of multiple, redundant, and diverse cloud components satisfies one of the key goals of the Systems Engineering Research Center Research Topic document, Part I (Section 4).



Legend for Figure 3:

A) Openstack VM & Microsoft Hyper V machines exchange video stream meta data every 60 seconds as a metadata comparison/sanity check of the data and of the current processing, including heartbeat monitor

B) One video stream is of a camera trained on the operator, as a check of current and valid operational processing

C) Video inputs include UAV FMV or infrared streams, the many building video or IR streams, the camera mounted on the RADAR dish (with compass overlay) video of the entry gate, and even the video of the AIMES or UAV operator(s) as described in b), the fence line(s), the UAV FMV/IR video, etc.

D) These server clouds house the video streams from the cameras (and other sensors) from around the building. If these servers are also used for more general data storage (especially if giving access to network users who gain access via a widespread network; perhaps SIPREnet, etc.), then each server cloud is protected by disparate firewalls (from 2 vendors) fronted by a load balancer (not shown here).

These few paragraphs discuss the many overlapping components involved in the process of detecting an approaching person or vehicle (an employee or authorized visitor) as well as the authentication and authorization of that person as they seek to approach the facility, seek to gain access to the building, and also to (ultimately) enter an internal laboratory:

Use Case: The overhead UAV and the RADAR atop the building detect the employee's car long before the seismic sensors detect the approaching vehicle, but all will verify the car stops at the parking lot outside the gate. Depending on the sophistication of the seismic sensors, individual vehicle signatures could be compared to a database of known employee cars OR the 32 bit ID codes from the vehicle TPMS (Tire Pressure Monitoring System) could be checked for authentication/authorization reasons (although new and rental vehicles will be 'first time events' for the system, such sophistication could be utilized for 'early warning' purposes). Biometric authentication (with a matchup with the RFID signature radiating from the employee badge and human guard backup) occur at the gate. Once clearing the perimeter gatehouse, the employee must authenticate again (with a different biometric parameter) at the entrance to the building within 2 minutes of passing the gatehouse. The seismic sensors around the building perimeter ignore the approaching human if the stride matches the employees' registered characteristics. Depending on sensitivity, authentication/authorization occurs once again before access is granted to internal labs (using one of the previously used biometrics, randomly sequenced).

A Simplified Model of the Previously Described Scenario (as Part of the Technology Demonstration):

Instead of using a fully scaled EXTERNAL scenario as described above, the team set up and monitored a much smaller 'internal' space in the OMERF building here at UVa and used that to create the model.

This smaller system uses two sets of sensors mounted on four (4) pods that are deployed around a center table that represents the building under protection ("the facility") as shown in figure 4:

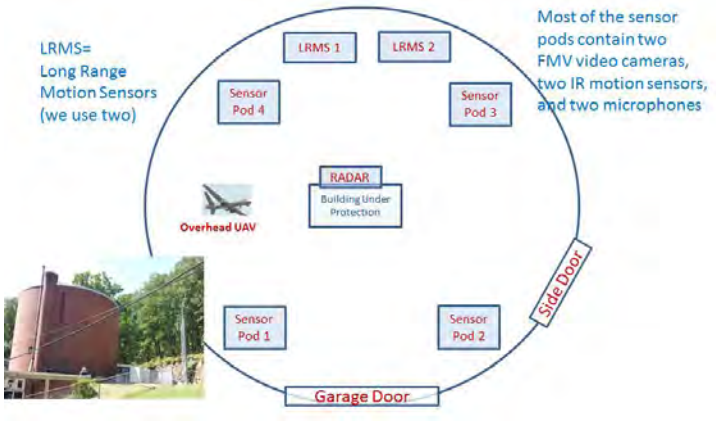


Figure 4: Physical Layout of the Facility and the Various Sensor Pods

As shown in Figure 5, each of the two sensor suites (an A side and a B side, on the various 'pod' platforms) would normally include ~8 cameras fed into AIMES (plus one additional feed from an overhead UAV camera), 8 motion sensors in the room and 8 acoustic microphones to pick up sound. The cameras, motion sensors, and microphones are mounted on the same platform, installed at a high enough level to be out of reach for anyone on the floor. The system would also have a long range motion detector based on RADAR and two smaller, less expensive Long Range Motion Sensors (LRMS) that are set up at the far side of the circular room and can 'sense' beyond the garage door (if it is open to the outside), and these LRMS sensors also turn on 4 floodlights to light the space (very useful if the room is dark). This RADAR and two (2) smaller LRMS can sense if/when someone walks through ANYWHERE the space and even outside, beyond the big garage door and into the parking lot and serve as long range sensors. The team intentionally made Pod 1 a pod with only ONE sensor of each type to eliminate dual redundancy on this pod and to facilitate our ability to attack the system to test the remaining sentinels (explained later). In other words, pod 1 was intentionally weakened. A RADAR system supplements the ability to detect and track intruders.

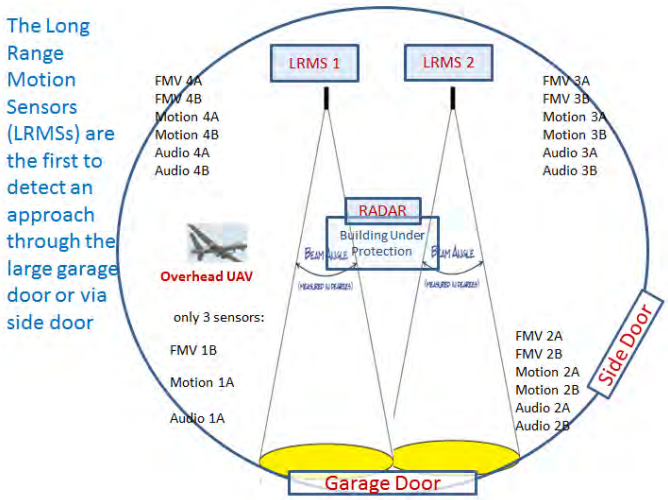


Figure 5: The Building Under Protection, the Sensor Pods, the Sensors, and the Physical Layout

Normally, the sensors would be close to the building under protection and would face OUTWARD to detect approaching intruders; but for the 'small space' ('interior') model established here, the pods are set up some distance away from the center table and faced INWARD as shown in Figure 6, below. As mentioned before, this scenario uses the internal spaces that we already have access to as part of the upper level of the OMERF reactor room, and the system can monitor the output of the sensor platforms in the conference room on the middle level, greatly simplifying the overall task.

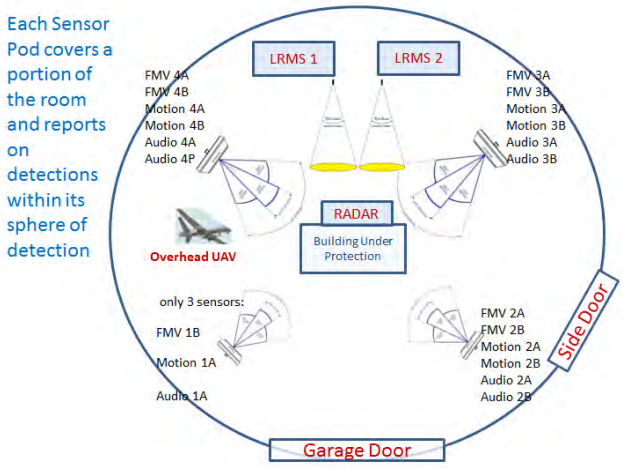


Figure 6: In the Simplified Deployment Scenario, all Sensors Face INWARD to Monitor the Space

Figure 7, below, shows typical pod sensors utilized by the subsystem, which was named the *Base Defense* System. This subsystem ingests the outputs of these four (4) sensor pods, the Long Range Motion Sensors (LRMS) and the RADAR output and makes decisions based on what these sensors are reporting.

The main personnel access door (the 'side door' in Figures 4, 5, & 6) retains its simple PIN entry code, but four (4) sensor platforms are mounted in the room, up high, and each sensor platform contains two sensor packages an "A" side sensor package and a "B" side sensor package. The only exception to this is that platform 1 has only ONE set of sensors. Platform #2 has two sets of sensors, but no

Sentinel. But normally, each sensor package contains a camera, a motion sensor, and an acoustic microphone that are connected into either the AIMES system or into the MultiSentinel device. The cameras are set up in a grid with 'crossing' fields of view that overlap each other; but for this simple scenario, are not electrically moveable. Future (and more complex) versions can include moveable, operator controlled cameras that movable/steerable, and this can serve to INTENTIONALLY trip the nearby motion or acoustic sensors with the motion of the camera(s) as a working test of the system.

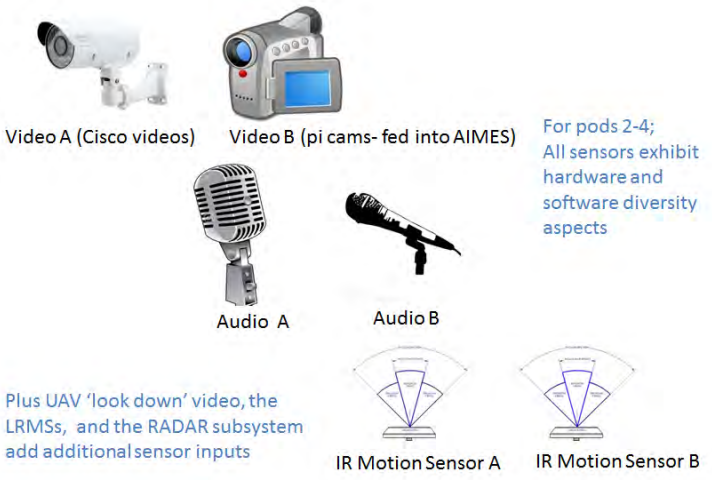


Figure 7: Typical Pod Components ('dual' A side and B side sensors, except for Pod 1)

In the simplified deployment scenario, some sensors report their output over a wireless link and other sensors are connected via CAT 5 (Ethernet) cable to reduce the chance of a wireless 'jamming' event impacting ALL sensors. In addition to this first sensor suite (Sensor Suite "A"), the team implemented a 2nd sensor suite of sensors, mounted on the same platform, that feed into the network and become part of the sentinel monitor backup/redundancy/comparison done at the operator station for improved 'system aware' functionality. Similar sets of sensors, from different manufacturers, feed over a different links to the controller console help insure that the same supply chain interdiction efforts will not be successful for BOTH sets of sensors and also insure non-interrupted service due to the dual nature of the system setup and data delivery mechanisms.

In a real deployment scenario, the sensor platforms would also have battery backup in case of total power failure and each platform on their own electrical power circuit, but (to save money) the team merely used Uninterruptable Power Sources (UPS) that UVa already had made available to the team (and we did not have to purchase). All of these aforementioned sensors comprise the *Base Defense* subsystem, as described in the next section (Appendix 2 also provides some operational details). This Base Defense subsystem also works with other components (AIMES and the RADAR display, for example) to provide additional capabilities and situational awareness.

The *Base Defense* Subsystem (Components and Function)—Including AIMES:

The output of the *Base Defense* subsystem is shown on the middle screen (with accompanying screenshot) in Figure 8, below. The *Base Defense* screen is merely one of several displaying all of the activities going on in the monitored environment. Other displays include (from left to right), the Cisco video cam feeds (see figure 9 for a screenshot), the AIMES subsystem display (showing video from the PiCams on each pod, see figure 10), and the RADAR display (see Figure 12 for a screenshot), and

(finally) a screen that monitors the real time execution of processing threads of the *Base Defense* System so that the operator is relatively sure of viable system operation (no screenshot provided).

The *Base Defense* Display shows messages as they are received from the various sensors on the four (4) pods. If a message is received, the operator should check out what is going on by using the two video displays on the far left (using 2 separate video sources) The Virtual Sentinel monitors the base defense messages, the LRMSs, and the RADAR for activity that is NOT being reported to the *Base Defense* system

Fundamental to the success of these various subsystems is the underlying operational details concerning the Sentinel components that detect and (as a minimum) alert the operator that failures have occurred, errant operation detected, or that the system has been attacked in some manner; this discussion (for both the Base Defense and the AIMES Sentinels) is contained in a later section.

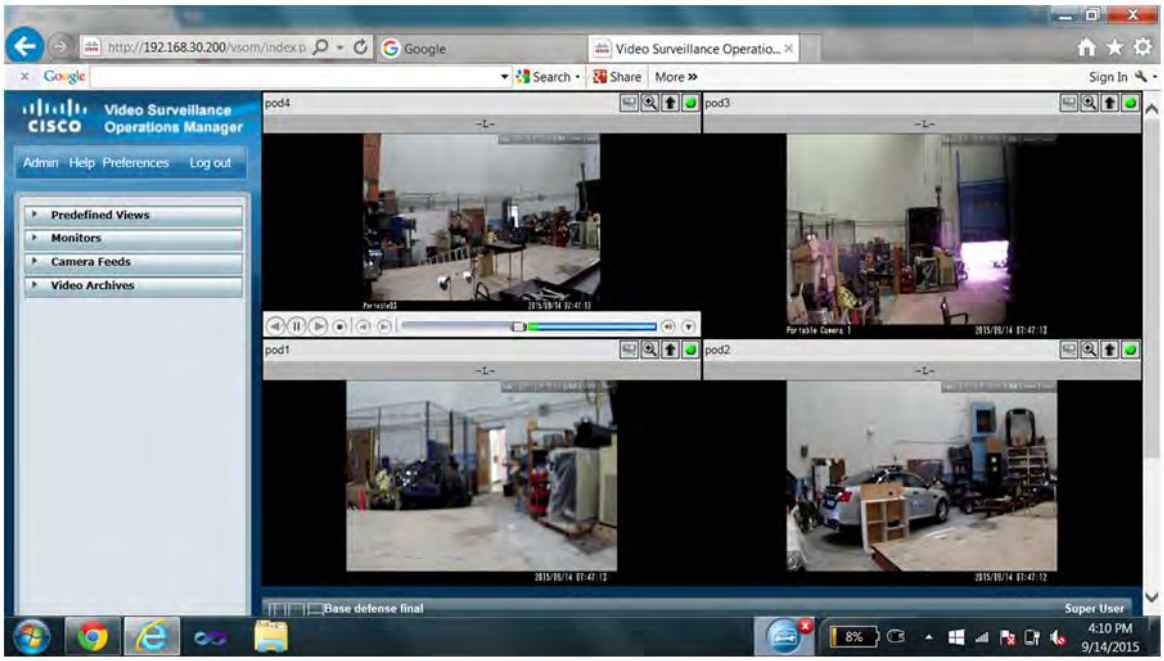


Figure 9: The Display of the four (4) Cisco Videocams, Monitoring the 'Building Under Protection'

The AIMES Subsystem from Leidos

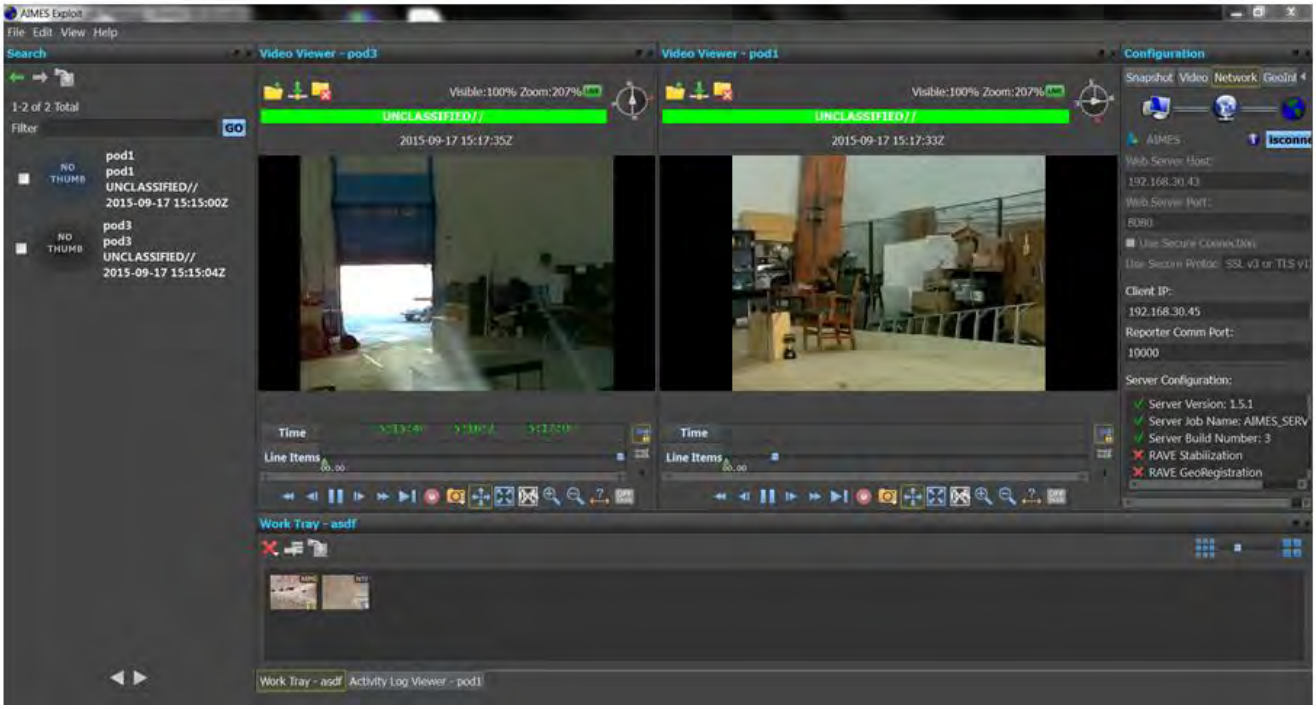


Figure 10: The AIMES display of the Monitored Space, and Promotional Brochure in Figure 11

Figure 11: AIMES Promotional Brochure, showing a Military Intelligence Use Case

Not included in this discussion or these screenshots is the AIMES Sentinel functionality, which will be covered in the next section This next section also discusses ways in which the subsystems were 'cyberattacked' and what the sentinel components did to protect the system or to alert the operator.

The AIMES Video Replay Attack (and a Sentinel Solution that Stops the Attack)

This section outlines the demonstration performed (on Sept 18th, 2015) of the AIMES video replay attack and the system aware protection measures put in place to detect such an attack.

Video Replay Attack

As summarized in Appendix 3, the video replay attack to the AIMES system replaces a current video feed with an archived video loop prior to an operator connecting to that feed. This situation may seem unlikely, but after gaining further understanding of an operator's role (disconnecting and reconnecting to different video feeds is a common operation in the field), one can gain an appreciation for how this may occur. This attack normally gives no obvious indication to the operator when they connect to the feed that the system has been compromised and that a bogus feed is being viewed. This was one of our main focuses when planning the approach; attacks on the system should not be of the denial of service variety (which is easy to detect), but more hidden and subversive in nature—making it much more difficult to detect. On the operator's screen, the correct timestamp, feed name, and mission name are all displayed-- giving no reason to suspect foul play.

System-Aware Protection to Stop the Video Replay Attack

After studying the system before and after the implementation of cyber-attacks, the team realized that very little information that is passed from the PICTE processing unit and the AIMES Exploit Client is presented to the user in an informative way regarding verification surrounding the feed, or (more significantly) important information gathered in PICTE was not transferred at all. Thus, the *system aware* approach to security becomes crucial to detect these attacks. This approach made use of the idea of a 'sentinel' being able to gather information from previously segregated components of the system and to use this information to form a decision on the state of the system.

In this specific application, the AIMES client, server, & PICTE comprise these segregated components. Each component contains information on the current active feeds and their associated metadata (including IP address and port). On the AIMES Exploit Client, this information was dumped into log files recording each connection to an active feed. On the PICTE processing unit, this information was stored locally (this will prove central to the design decisions) in an XML file for each feed instantiated in PICTE. On the AIMES server, there is a table that contains a list of active and inactive feeds. In short, all three sources were parsed for the IP address and port of the active feeds and sent via rabbit to a program on the sentinel that blended and compared all of this data (collected every five seconds approximately), allowing the system to continuously monitor the consistency of the data.

A file on the system was crucial because it is the only one (out of all three) that had the least networking vulnerabilities. The information from the PICTE was determined by the stream going to the PICTE feed, but this was not where the attacks were directed. The attacks were launched between the Client and PICTE, so the information dumped into the *Exploit Logs* could be invalid because this information came from PICTE's status update. This status update was what was spoofed and utilized as the point of attack, simultaneously forcing the Exploit logs to archive this spoofed data. The team therefore determined that because the information on the PICTE machine went

through no network traffic to be archived, that this data would be the "golden standard" and the least likely to be falsified. Differences from PICTE's XML file containing the IP address and port of an active feed would be recognized by the sentinel program and an alert sent to the client operator. This functionality was previously unavailable without the sentinel component, a perfect example of how traditional cybersecurity's tactics of putting up 'tougher and higher walls' may be less effective than combining the data from various sources to detect anomalies in a more 'systems aware' fashion.

Throughout many attempts to determine the best way to ensure validity of the system functions, this idea of using segregated data and making decisions based on the known functionality of the system (compared to the current functionality) was the most successful. It is also clear how this idea could be expanded to detect other anomalies in the AIMES system, as well as other systems with a similar distributed client/ server/ processing relationship. Before the final presentation, this *proof of concept* was demonstrated on a raspberry pi video stream, as well as the UAV video feed. Both systems could undergo a video replay attack, both attacks can trigger a detection by the data blending program, and this program presents the AIMES Exploit Client with the correct IPs and ports for detecting & rectifying the situation. See Appendix 3 for a few of the details on how to configure the protective mechanisms & to execute the various replay attacks; but for the purposes of this report, precise details of the attacks have been removed. Other Sentinel capabilities were demoed during the Sept 18th meeting at UVa, but all were along these same goals—detect the attack and then deploy/invoke countermeasures to stop the attack.

The Local RADAR Subsystem ('local' to sensor pods)

The project purchased a multi-module RADAR as part of the system (and this functioning splendidly, but only as a single module that detects and reports range to the target).

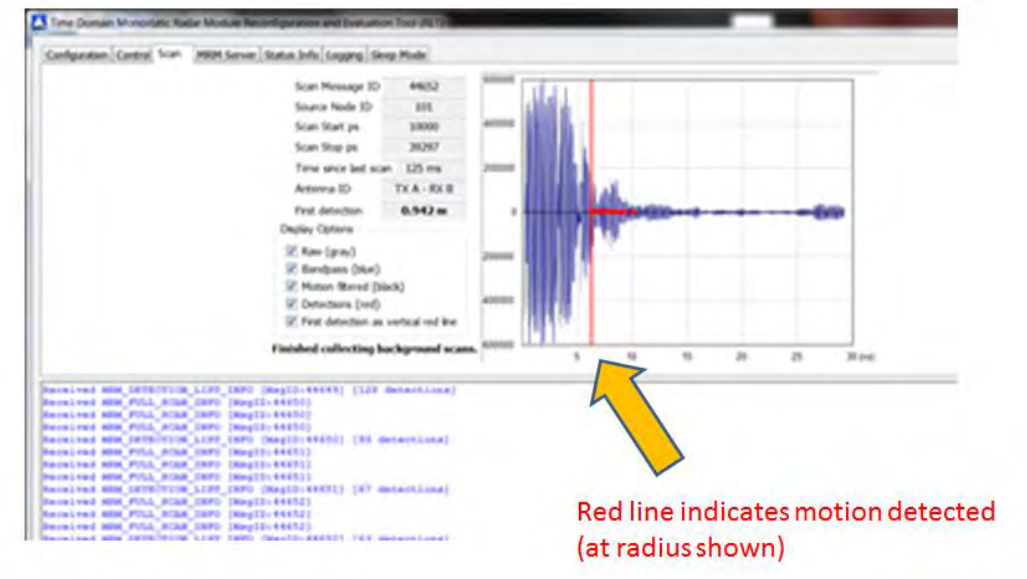


Figure 12: The RADAR Display Showing the Distance at which an Intruder is Detected

Use Cases in Normal Operation (Present and Potential Future Ones):

The following paragraphs outline six (6) possible "intruder detection" use cases for this *Base Defense* system:

Detection Scenario #1 **Pedestrian** traffic inside the Reactor Room **WITH** early detection via the RADAR or the Long Range Motion Sensors (LRMS)

Background: A heavily protected building (perhaps a nuclear reactor building, a weapons lab, or a high security prison) is represented by a small table in the center of the large, round room that is called "The Reactor Room" in the OMERF building at 675 Old Reservoir Road. The area around this small table represents the surrounding fields and airspace that is loaded with sensors to detect unauthorized visitors ("intruders").

Scenario Abstract: A person walks into the open garage door of the OMERF Reactor Room, triggering responses not just from the early warning RADAR, but also from the two (2) Long Range Motion Sensors (LRMSs) as a preliminary alert that warns the system's operators (and also turns on some spotlights so that cameras can record activity in this sometimes darkened room). LRMSs and RADAR should be triggered first and provide the preliminary warning, but the PIR motion sensors on the various pods (pods 1, 2, 3, and 4) should pick up the intruder(s) as they proceed deeper inside the reactor room space (the activated pods depend on direction chosen). Tracking of the intruder is initiated and responses are considered, including responses to events that don't seem reasonable (and thus may be an attack).

Scenario detail (Preliminary Alert): As a person approaches the open garage door that is on the perimeter of the large reactor room, the Long Range Motion Sensors (LRMSs) and possibly the early warning RADAR device pick up this motion and send messages to the *Base Defense* System. As for the LRMSs, BOTH sensors should detect the intruder at about the same time, unless the intruder is moving in a manner that tends to favor one sensor's view field as opposed to the other sensor's field of view--but this should be rare. If only ONE LRMS (and the RADAR) senses movement, the non-responding LRMS sensor should be checked for proper operation and the results gained thus far (the RADAR and the other (functioning) LRMS) be used as a successful two-out-of-three voting scheme and the preliminary intrusion alert deemed valid, albeit suspicious due to the missing LRMS alert.

Scenario detail: After a preliminary alert is received by *Base Defense*, operators assume a higher alert posture and actively monitor the nine (9) camera feeds that comprise the total video system. The total video system is comprised of the CISCO VSOM display (for the four (4) CISCO cameras), the four (4) Raspberry Pi cameras (called 'Pi cams'), and a 'lookdown' view of the entire from an overhead UAV that hovers above the protected space in the center of the reactor room. Unlike the other sensor platforms, this UAV is free to move about the airspace to provide unique and close up camera views of suspicious events or activities and is especially equipped to do so in this enclosed space. Also, this UAV could carry a payload of disposable ping-pong balls that simulate grenades or air-to-ground missiles that attempt to stop (or disperse) an approaching intruder that appears malicious. In the most severe system response in our simulated environment, the unidentified and unauthorized intruder could be bombarded from above by the UAV with something as small as ping pong balls.

After the preliminary alert is issued and the intruder progresses inside the reactor room, the various sensor pods should detect this activity and provide alerts to the *Base Defense* System. If the intruder enters the room closer to the left hand side (as viewed from outside the garage door), this should favor the "Pod 1" sensor pod. Pod 1 is the least capable pod because it is equipped with only one camera, one motion sensor, and one audio microphone, each fed into a Raspberry Pi Single Board Computer (SBC). Pod #2 has these devices PLUS an additional Arduino SBC that has an additional complement of these motion and audio sensors and a stand-along CISCO IP video camera that feeds

into a VSOM plug in that can be viewed from any Internet Explorer (IE) browser with this VSOM plug in. Pods 3 & 4, the most capable pods, have this dual compliment of sensors PLUS a sentinel device that seeks to help determine faulty operation and/or malicious or attacks (with bogus inputs) on the system. Two pods send their messaging back to the *Base Defense* System via a wired network, and two pods send their messages back over a wireless network. One SBC sends messages back via the Rabbit MQ messaging service, while the other SBC sends messages back in a standard UDP/IP format.

As messages are generated from the various pods (with Pod #1 the least capable and easiest to 'spoof' and pods 3 and 4 the most robust with their dual sensors and sentinel device), operators are able to determine proper sensor operation based on the messages received. The early warning RADAR image of the mobile intruder can be used to confirm sensor pod message results and to confirm actual location of the intruder(s). The *Base Defense* operators can command the overhead UAV to move to an area where overhead monitoring can be enhanced. As a test of the system that triggers all sensors, the overhead UAV ("defense UAV") can be commanded to fly lower, into the view of the LRMS sensors, the RADAR, and around to each of the four (4) pods to exercise/trip each of the PIR motion detectors and microphones. The video from the pod cameras, the sensor messages, the LRMS responses, and the RADAR returns can verify that the UAV is indeed present and is the device creating all of these triggers. The aerial intrusion of the UAV into the protected space should produce a large number of alerts and system responses which can be used for verification purposes.

Detection Scenario #2 **Pedestrian traffic** inside the Reactor Room **WITHOUT** early detection via the RADAR or the Long Range Motion Sensors (LRMS)

This scenario is the same as above, except the intruder is detected by the sensors or cameras on the pods ONLY (pods 1, 2, 3, and 4) and tracking is initiated and responses are considered. The LRMS devices (and even the early warning RADAR) may or may not be triggered, depending on the source of the detected motion or sound; the source of these alerts may be out of their view. But, the remaining system components (pods 1, 2, 3, and 4) and the overhead UAV have plenty of detection and warning capability to spare.

Detection Scenario #3 **Aerial Intrusion** into the Reactor Room Space via the outer perimeter (**WITH** early detection via the RADAR or the LRMS)

Same as Detection Scenario #1, except that the intruder is a UAV flying into the Reactor Room spaces (from outside) via the big garage door, and the detection/responses should unfold just like the previous scenarios, albeit that the audio sensors may be more actively triggered due to the noise of the UAV. The inputs to the audio microphones may have to be desensitized (or even filtered out) if the noise of the UAV is too loud and it triggers all pod audio devices simultaneously.

Detection Scenario #4 **Aerial Intrusion** into the Reactor Room Space from above (**WITHOUT** early detection via the RADAR or the LRMS, but RADAR is the 1st to see the intruder)

In this scenario, the UAV is hovering ABOVE the protected area, high in the Reactor Room, initially unseen by any of the sensors protecting this space. As the UAV descends (to either attack or to probe defenses), the various sensors are triggered—probably the LRMSs or the RADAR initially, followed by other sensors (such as the PIR motion detectors and the audio microphones) and then visual tracking can be accomplished via the cameras. An aerial intrusion scenario such as this should be performed

on a periodic and planned basis to insure that all sensors are indeed functional and that the operators and the sensor systems provide expected response(s).

Detection Scenario #5 (future growth) Seismic/vibration sensors to detect underground tunneling from below, possibly from across the driveway as shown in figure 16. The project has not purchased sensors to do this; but it is identical to the PIR motion sensor scenario and response and is an example of another useful sensor to use in the array of sensors.

Detection Scenario #6 (future growth) Seismic detection along a fenced perimeter using fiberoptic cables; the perturbations in the light pulses indicate seismic activity that can be easily characterized as human versus animal versus mechanical.

This used to be the purview of very costly systems ONLY, but there are several commercial companies that now have a system that trenches long strands of fiberoptic cable just outside a fenced perimeter, and any seismic activity in the area perturbate the optical pulses in a manner by which human, mechanical, or various animal activity can be identified and characterized. The software is sophisticated enough to detect (and announce) whether or not the source of the seismic perturbations are human, mechanical (as well as vehicle type), or from some species of animal.

System Development and Research Goals

Providing Artifacts in Compliance With the SERC Research Topic Document

The majority of the work done on this effort was accomplished during the summer months, when the team had as many as 2 full time staff members and 7 student employees engaged; the team was wise to create a 'midsummer' status report paper showing Professor Horowitz what had been accomplished and what the team was working on. This entire status report is too lengthy (and is available upon request), but Appendix 2 discusses general accomplishments

Early in the project, the team was divided into two sub groups working on different aspects of the system. One team focused on the sensors, the camera interfaces, and the messaging protocols, while the second focused on the AIMES subsystem from Leidos. A summary of the project status from each team (as of midsummer) is included in this document as Appendices 2 and 3; Appendix 2 from the 'sensor' team and Appendix 3 from the AIMES team, have been greatly shortened from their original length to remove details of how to perform certain attacks--but either document is available in its original form upon request.

Even at this midsummer date, the teams had identified attack surfaces of their respective subsystems, possible attacks to attempt, and potential sentinels to employ as defenses. Work began in earnest to meet the goals specified in Part I, Section 4 of the SERC Research Topic document. As a summary of that early activity, it is sufficient to say that the 'sensor' sub team had experimented with video streaming (from the Raspberry Pi's), researched and purchased the necessary sensors to install into the various pods, set up and started using the RabbitMQ/UDP server for messaging back and forth between sensors and computing units (as well as the *Base Defense* application), and constructed a few cyberattacks and defenses. A basic schematic of the networking of these messages is shown in Appendix 4.

Likewise, the AIMES sub team had mastered the various features of AIMES and also generated some exploits of that subsystem, as well. They learned how to create a video 'replay' attack (and a defense

thereof), they performed an attack on video quality, and then learned how to create a video 'metadata' attack (and defense). All of this is briefly summarized in Appendices 2 and 3, respectfully, but precise details of the various attacks are removed for reasons of sensitivity. There are general and generic references to attacks and attack types, but the details of how to launch attacks have been intentionally removed from this document. The AIMES team came up with a particularly creative name for their non-cloud protective mechanisms, referring to this protection functionality (and the GUI interface) as being the AIMES "Sidecar", and this functionality is described (in general terms) in Appendix 3. It is clear that by mid-summer, the system had been built and project was indeed in great shape and ready to move on toward completion

Displaying Sensor Outputs to the *Base Defense* Operator(s):

Throughout the project, careful thought was given to how the outputs of these various sensors would be displayed to the operator(s) of the system; and indeed, this topic can frequently result in very lengthy and convoluted *Human Factors* debates –so the team elected to pursue a 'middle ground' in terms of sophisticated visual displays of system status and sentinel results versus simple text message outputs to the screen (and to the logs). And not just for the sensors in the pods alone, but also of the other miscellaneous components of the system (the Long Range Motion Sensors (LRMS), the RADAR, the two video camera subsystems, the video output of the UAV, and other sensors that might be added as a phase of future growth). Some of these could be incorporated into the basic *Base Defense* screen, but some of these would occupy their own full size computer screens (particularly the RADAR and the video streams from the Cisco cameras and the SBC PiCams).

To meet and exceed the goals in Part I, Section 4 of the SERC Research Topic document, constant experimentation was performed as the system gradually grew in size and complexity over the course of the project. The constant research into new sensor product technologies coupled with the rapid implementation of suitable devices was a powerful combination that kept the team busy and engaged. And although there is a lot of evidence of 'compliance' sprinkled throughout this document, the team has summarized the compliance elements in the following table.

Table 1 (below) shows a shortened version of the goals of Part I, Section 4 of the SERC Research Topic document, along with the system components & features that meet or exceed the stated goal

Develop sentinel design patterns to support detections across the architecture	Design decisions later in the development cycle supported and enhanced earlier design decisions by virtue of additional layers of redundancy, component diversity, data checking, and other *system aware* techniques.
Monitor and verify sentinel network performance	Tests were performed to insure that intruders were detected in a rapid and consistent manner across the network
Access the scalability of the multi-sentinel concept	From the documentation presented, it should be readily apparent that successive layers of sentinels could be added as layers (or quantities) of sensors are added.
Develop performance requirements for a sentinel networking concept	The most important of the requirements would be that the network (wired or wireless) supports the quantity of messages being sent; insuring timely delivery. Network monitoring tools such as *Wireshark* were used to make sure the network was not saturated with traffic. Also, Ethernet link lights were monitored to insure a minimal number of 'collisions'. Excessive collisions are a sure sign of an overworked network.

Identify areas where a rapid prototyping effort could be proof of concept in FY 16	The extensive use of products like the Raspberry Pi and the Arduino SBC have paved the way for more extensive use (to include the use of our more popular and associated sensors)

Table 1: Part I, Section 4 Goals and the Research or System Components that Meet These Goals

The end result, for the *Base Defense* display, is shown in figure 13 (below):

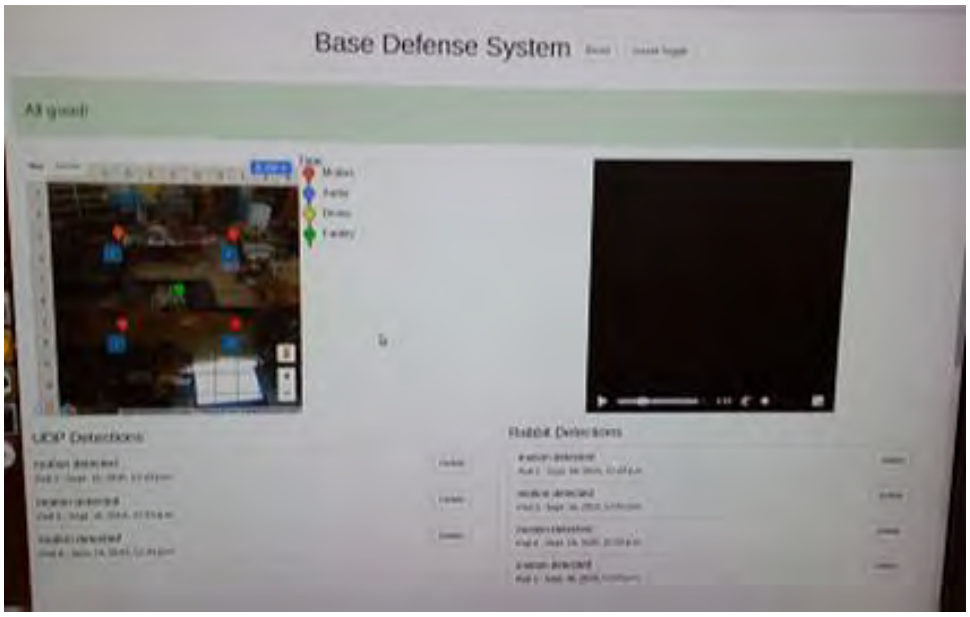


Figure 13 Base Defense Subsystem Display

The team felt that this was the best compromise between a fully 'visual' display and one that would merely be full of text messages. On the left, a snapshot of the 'Building Under Protection' (a table in the center of the OMERF Reactor room space) has color-coded icons that are dropped onto the screen at or near the spot where a pod sensor has detected either sound or motion, and the resulting UDP server message(s) and/or the resulting RabbitMQ message(s) are printed out on the lower part of the screen. A spot for the UAV's 'look down' video input was also designed in, shown in this screenshot as a black field on the right hand side of the display. Ideally, any and all sensor responses from an intruder would result in icons on the left side picture, message printouts for both the UDP and RabbitMQ servers (signifying that both "A" and "B" side sensors were triggered by an intruder); and if the source of the motion or sound is in the view of the UAV's look down video, the person (or animal) would show up there on that video. Likewise, if the source of the sensor detection was in the view of the Cisco cameras or the Raspberry PiCams, that would be visible on the adjacent display(s), as well. Appendix 4 shows an example of message flows between the various components.

The team also felt that some sense of real time 'activity' was warranted; in other words, was the sound or motion just detected a one-time event, or have there been several OTHER detections recently that should 'raise the level' of operator awareness to possible intruders?? If only one sensor (or less) had been triggered, the result would be the green stripe across the top of the screen shown in figure 13. In this snapshot, NO recent sensor detections have occurred, so the *Base Defense* subsystem has determined that there seems to be no intruders in the immediate area and the stripe is green with an "All Good" text message displayed. In cases where there are a few sensor detections, but few and far between, this green strip will turn YELLOW in color (with a different text message),

signifying that the operator(s) should maintained a heightened state of awareness and alertness, as there may be unofficial (and uninvited) activity in the area. In the event of a lot of sensors being triggered, this yellow stripe turns **RED** and flashes to make sure the operator is paying attention to the signs of a certain intruder approach. A more detailed ALERTCON scheme was proposed that gave specific weights to certain sensors (some sensors have more 'credibility' than others) and this scheme is detailed in Appendix 5, but was determined to complex for this simple system (but may well find a home in a more complex and production-ready *Base Defense* subsystem).

In a somewhat similar manner, messages were contemplated for situations where a sentinel was triggered because the system was not behaving the way it was designed, indicating that either some sort of random component failure had been detected OR that the system may be under direct cyberattack. Further discussion of sentinels and sentinel messages are being deferred until the next section, but the reader should be able to envision this situation. These messages are almost limitless in scope and complexity, and are bounded only by the imagination, but a few examples are shown in Appendix 6 as potential Sentinel Alert Messages (SAMs). System aware sentinel components were developed for two of the four sensor pods and for the AIMES subsystem, as well.

Sensor Pod Sentinels & Protection Against USB Device Insertion Into the Pod SBCs

Another aspect of this project was to implement *system aware* sentinels on two of the four sensor pods so that a separate, independent set of sensors could be used to perform a two-out-of-three voting scheme to determine if a sensor message should be sent off to the *Base Defense* subsystem. The problem with having only two sensors is that if one sensor is triggered and the other sensor is not, then what should the system report, if anything?? Having a 3rd set of sensors, also routed through its own SBC (Single Board Computer), allows the system to implement a two-out-of- three voting scheme to settle the course of action. In many other systems (air transport-class autopilots on Boeing-type aircraft, for example), the outlying detection will be reported to the pilots in a *Minority Report* fashion; but other systems (such as the one developed for this program) will discard the minority result, unreported.

To help facilitate this function, a Sentinel web application was developed. Details are included in Appendix 7, but this application was modeled after the Base Defense web application and was very similar to it, but only applicable to pods 3 and 4, not all four pods. This decision was partially to save hardware costs (and complexity), but also allowed for the team to demonstrate some attack examples without being thwarted by all of the extra monitoring and reporting that would result from four fully capable pods. In other words, the system had been weakened for demo purposes.

Appendix 7 also has details about a sentinel monitoring function that guards against the insertion of a thumb drive into the USB ports of the various SBCs in an attempt to replace the operating systems of these devices and to interfere with the normal functions of these SBCs who are either performing the sensor reporting function OR sentinel monitoring duties.

Using a UAV as a Sensor, a Roving Video Feed, and a Sensor Trigger for Self-Test

Early in this project it was hoped (even ANTICIPATED) that a small UAV would be a big part of the overall Base Defense system. Figures 1-6 show a UAV flying high above the building under protection, providing a bird's eye view of the landscape and playing prominently in this scenario. And for a 'real' deployment of such a system, the team certainly feels that is certainly the case.

One should easily see how the UAV could patrol the overhead skies, capturing video of the area and spotting approaching intruders from long distances. In addition, the ability for the UAV to make passes over the building and to appear in the video feeds of the sensor pods, and perhaps even triggering the pod's audio sensors, both make for useful 'tests' of the system for proper operation (was the UAV 'seen' in the video at the expected location?? Were audio sensors tripped or the UAV detected, where expected, by RADAR??). All of these are typical *systems aware* techniques that are useful for verifying correct system operation and performance, usually called *data verification*.

However, when the system is scaled down to a small, confined space INSIDE a building, several problems emerge. First on the list is just that—the confined space. The team discovered that it was very difficult to fly ANY UAV in the confined space of this interior reactor room; not just due to the small size, but also due to the fact that the UAV cannot get valid GPS signals inside the building. This was an unexpected anomaly; the team underestimated the effect on the ability to conduct stable flight inside a building without GPS. So in addition to confined space without GPS, the next problem was the NOISE of the UAV. Even small, 'allegedly quiet' UAVs made too much noise in this confined space and constantly triggered not just the audio sensors on a constant basis, but also the motion detectors and RADAR. And although the images in Figure 14 make it appear successful UAV flight is possible, the reality was that it took superior operator skill to keep the UAV from crashing; and the image on the right side of Figure 14 shows the UAV perilously close to a Virginia State Police car that was in our space for a different research project at that time. Fortunately, disaster was averted in these flight test situations, but was always a very real possibility.

Figure 14 Attempted UAV Flights Indoors (with minimal success)

The team tried several means of enhancing the flight stability (including a greatly illuminated launch and hover area as shown in the left side of Figure 15), but minimal success was achieved. It was less of a problem later in the project, but early on the excessive clutter in this space was a factor in performing UAV operations. The right side of Figure 15 shows where pods 1, 2, 3, and 4 were eventually positioned and one can see the inordinate amount of clutter that was initially in this area at the start of the project, but was successfully removed later.

Figure 15 The attempts to Achieve UAV Flight in the Upper Reactor Room at the OMERF

Human Factors

Preamble

Over the course of this research project, a remote team member at AFRL (LtC Chris Gay) examined the current literature on trust and suspicion to acquire the necessary understanding of those broad disciplines and to inform the application of behavioral science to the operator/Sentinel cyber-attack response process. Based on his literature review, he believes the concept of suspicion as proposed by Bobko et al. (through Air Force Research Lab (AFRL) & Air Force Office of Scientific Research sponsorship) has potential to develop a more resilient operator. LtC Gay has begun to apply this theory of suspicion to the study of operator response to cyberattacks in the context of a cyber physical system, and he has developed a design of experiment concept to rigorously test the theory of suspicion and its relationship to operator performance against cyberattacks. This is second of the important research areas stipulated in Part I, Section 4 of the SERC Research Topic document.

Based on his progress in understanding the concept of suspicion, its potential implications to operator cyber response, and a design of experiments to empirically test it, the proposal for next year will focus on the experimental process necessary for data collection and analysis to answer the research questions. He will develop several scenarios in which the key components of suspicion (uncertainty and perception of malintent) will be manipulated along with several different combinations of cyberattack / no cyberattack, Sentinel aid correct / not correct, consequence lo / hi, and perception of consequence lo / hi. His results (if the proposal is accepted) will be reported on next year.

Human Factors Executive Summary

System Aware Cyber Security is an approach developed by the University of Virginia (UVA) which creates security architectures from a systems design rather than a perimeter security perspective. It incorporates defense mechanisms within the system to be protected such as: (1) diverse redundancy and defense in depth, (2) system configuration hopping, and (3) tactical forensics [1]. Although this security concept showed promise in several studies, the early research efforts focused on the conceptual, hardware and software development, leaving human factors issues associated with operating this technology to future research activities.

During the 2013-2014 DoD SERC effort [2], UVA and Georgia Tech Research Institute (GTRI) continued to advance the System-Aware Cyber Security concept through development of a secure system monitor referred to as a Sentinel. A Sentinel was designed to detect illogical and potentially damaging system behaviors, and it was mounted onto an “Outlaw” unmanned aerial vehicle for live flight-test performance evaluations. Additional research collaboration was also conducted in parallel between UVA and MITRE to examine the application of the Sentinel technology for operations of remotely piloted aircraft systems (RPAS), specifically the MQ-1/9, in the US Air Force (USAF). The UVA / MITRE team explored development of a methodology to design and test cyber-security Concepts of Operations (CONOPS) in the context of RPAS operations.

The goal of this effort was to assure minimal mission disruption in the event of cyber-attacks during military operations. To that end, a plausible RPAS cyber-attack scenario was designed to validate the Sentinel’s value. The scenario was modeled by MITRE’s Resources for Early and Agile Capability Testing (REACT) lightweight simulation lab to examine the value of Sentinel technology with a cyber-

attack response CONOPS in Air Force operations. The REACT model simulated the RPAS environment, the Sentinel component, and cyber-attack scenario to seek further confirmation of the Sentinel's effectiveness and allow for iterative development of effective cyber-attack response. A tabletop exercise based on the REACT cyber-attack scenario model was conducted with USAF RPAS operations subject matter experts. Their feedback validated the concept of using REACT to concurrently develop the Sentinel user interface and a cyber-attack response CONOPS. The participants knew of no other program looking at cyber-attack from this human / mission perspective, and they thought Sentinel-like technology and clearly defined CONOPS would be key to detecting and mitigating the operational mission impact of cyber-attacks.

While the REACT tabletop exercise results demonstrated great promise for incorporating a Sentinel type capability into future RPAS, evidence from this effort highlighted the inability of RPAS operators (without a Sentinel aid) to detect cyberattacks when unwittingly subjected to them, which lead to decreased mission performance and/or mission failure. Even with the Sentinel aid, the operators expressed concerns. The following statements were made during those experiments:

"We are trained to trust the systems provided to us. …If a Sentinel reports a cyber event and helps correct it, how do I know there won't be another attack that could take over the aircraft or the fire weapons? …I would likely return to base after first cyberattack and not continue the mission."
Creech Operators, August 2014

The operators did not appear to become suspicious of any malintent during simulated cyberattacks. Given the increasing likelihood of operations in a cyber-contested environment, these findings present a serious concern and motivation to search for potential solutions to operator awareness of and response to cyberattacks.

The research is motivated from the findings in the REACT tabletop exercise and explores the more basic human dimension of cyberattacks. Specifically, it builds on Bobko, Barelka, and Hirshfield's [3] work on suspicion by investigating the effects of operator suspicion (trait and state-level) on the detection of cyberattacks. The research aims to address several key questions related to operator suspicion and its potential role in detection of cyberattacks such as:

- Is a suspicious operator more capable of detecting cyberattacks?
- What trait-level (individual) attributes are related to one's ability and disposition to become suspicious?
- What is the relationship of operator suspicion to detection of cyberattacks in an operator / Sentinel team?
- What are the effects of consequence and operator perception of consequence on operator suspicion and associated outcomes?

This paper discusses the foundational work conducted to achieve these research aims and concludes with a discussion of a design of experiments to study the relationship between operator suspicion and detection of cyberattacks.

We thank Prof Philip Bobko, Gettysburg College, for his assistance, perspectives and feedback regarding application of his state-suspicion theory and model.

Introduction: Exploring the Role of Suspicion in Detection of Cyberattacks

Motivating Scenario

During the 2013-2014 DoD SERC effort [2], UVA and Georgia Tech Research Institute (GTRI) continued to advance the System-Aware Cyber Security concept through development of a secure system monitor referred to as a Sentinel. A Sentinel was designed to detect illogical and potentially damaging system behaviors, and it was mounted onto an "Outlaw" unmanned aerial vehicle for live flight-test performance evaluations. Additional research collaboration was also conducted in parallel between UVA and MITRE to examine the application of the Sentinel technology for operations of remotely piloted aircraft systems (RPAS), specifically the MQ-1/9, in the US Air Force (USAF). The UVA / MITRE team explored development of a methodology to design and test cyber-security Concepts of Operations (CONOPS) in the context of RPAS operations.

The goal of this effort was to assure minimal mission disruption in the event of cyber-attacks during military operations. To that end, a plausible RPAS cyber-attack scenario was designed to validate the Sentinel's value. The scenario was modeled by MITRE's Resources for Early and Agile Capability Testing (REACT) lightweight simulation lab to examine the value of Sentinel technology with a cyber-attack response CONOPS in Air Force operations. The REACT model simulated the RPAS environment, the Sentinel component, and cyber-attack scenario to seek further confirmation of the Sentinel's effectiveness and allow for iterative development of effective cyber-attack response. A tabletop exercise based on the REACT cyber-attack scenario model was conducted with USAF RPAS operations subject matter experts. Their feedback validated the concept of using REACT to concurrently develop the Sentinel user interface and a cyber-attack response CONOPS. The participants knew of no other program looking at cyber-attack from this human / mission perspective, and they thought Sentinel-like technology and clearly defined CONOPS would be key to detecting and mitigating the operational mission impact of cyber-attacks.

While the tabletop exercise results demonstrated great promise for incorporating a Sentinel type capability into future RPAS, evidence from this effort highlighted the inability of RPAS operators (without a Sentinel aid) to detect cyberattacks when unwittingly subjected to them, which lead to decreased mission performance and/or mission failure. Even with the Sentinel aid, the operators expressed concerns. The following statements were made during those experiments:

"We are trained to trust the systems provided to us. …If a Sentinel reports a cyber event and helps correct it, how do I know there won't be another attack that could take over the aircraft or the fire weapons? …I would likely return to base after first cyberattack and not continue the mission."
Creech Operators, August 2014

The operators did not appear to become suspicious of any malintent during simulated cyberattacks. Given the increasing likelihood of operations in a cyber-contested environment, these findings present a serious concern and motivation to search for potential solutions to operator awareness of and response to cyberattacks.

The Problem

There is considerable effort ongoing to prevent or detect and mitigate cyber-attack on DoD networks and IT systems. Autonomous- and tele-operation of aircraft also represent an intrinsic vulnerability, or at the minimum, a possibility for adversaries to perform cyber-attacks for counter-control or

subversion of the military assets. Several prominent cases highlight that a cyber-attack could be highly sophisticated, evade detection and disguise its impacts while inflicting significant damage to cyber-physical systems or their missions. For instance, Stuxnet was a computer worm developed specifically to subvert the Iranian uranium enrichment program by sabotaging the rotational speeds of centrifuges while masking the erratic speed fluctuation from the operators. The result was severe damage to the centrifuges without any physical strikes on the facility. In essence, the digital technology enabling control of complex physical equipment can become a venue of attack to inflict physical damage on cyber-physical systems. Given the reliance on digital technology for autonomy and control of Remotely Piloted Aircraft Systems, cyber-attacks pose as much threat as physical attacks to military operations.

Given the sophistication and potential consequences of cyber-attacks targeting RPAS to thwart military operations, the need for a Sentinel-like capability is great. However, a technology solution in and of itself may not provide the level of security required. The technology itself is fallible. For instance, it may not be programmed to detect the latest emergent threat, or it could generate an alert to anomalous system behavior unrelated to a cyberattack, such as a maintenance issue. Ultimately, the human operator is the decision maker. The operator must determine in all cases whether the Sentinel aid is correct or not correct from all the available information. In some cases, the operator could conceivably need to override the Sentinel, and in other cases, the operator may need to intervene when the Sentinel does not. Therefore, performance against cyberattacks must be viewed from the standpoint of an operator / Sentinel team with emphasis on the operator's ability to accurately assess & respond to a given situation.

As previously highlighted, much work is being done on the technology side of the operator / Sentinel team. However, little is known about the behavioral sciences driving operator response to the cyber situation and the Sentinel aid. Consider again the following statements made during the previous experiments:

"We are trained to trust the systems provided to us. …If a Sentinel reports a cyber event and helps correct it, how do I know there won't be another attack that could take over the aircraft or the fire weapons? …I would likely return to base after first cyberattack and not continue the mission."
Creech Operators, August 2014

To achieve desirable level of resilience (i.e., minimal mission disruption), the RPAS community, or the military services in general, must address the following questions and issues:

The Human-dimension of Cyberattack on RPAS operations has not been studied.

Given operators are "trained to trust," what is the relationship of trust to detection of cyberattacks? Given the RPAS operators did not suspect cyberattacks (malintent) when unwittingly subjected to them, what is the relationship of suspicion to detection of cyberattacks? Is a suspicious operator more capable of detecting cyberattacks? What individual (trait-level) attributes contribute to an operator's capacity to become suspicious? What is the relationship of operator suspicion to detection of cyberattacks in an operator / Sentinel team? What are the effects of consequence and operator perception of consequence on operator suspicion and associated outcomes?

The UVA/MITRE team's preliminary investigation into operations involving RPAS suggests that military operators typically have no cyber-security preparation or role during missions. Most of the

cyber-defense in the military across all services appears to exist outside of mission operations in the form of information assurance programs and cyber command centers with the focus being on protection, not detection, remediation, and ensuring resilience. Military personnel conducting mission operations can be a main source of resilience in cyber-security because technological solutions by themselves have limitations, particularly in adaptation to contexts and response to cyber-attacks. The integration of operations personnel in the response to cyber-attacks requires substantial research and development effort from the organization, technological, behavioral sciences and training perspectives.

Human Factors Approach

In pursuit of the goals stipulated in Part I, Section 4 of the SERC Research Topic document, UVA partnered with the Air Force Institute of Technology (AFIT) to study this problem, which can be characterized as basic research in behavioral science regarding operator detection and response to cyberattacks. The UVA/AFIT team determined a phased approach to be the best way to investigate this issue. Thus, the approach developed to study the human dimension of operator cyber detection included the following high-level steps:

1. Review and application of trust and suspicion literature to the problem

We examined the current literature on trust and suspicion to acquire the necessary understanding of those broad disciplines and to inform the application of behavioral science to the operator / Sentinel technology cyber-attack response process.

2. Design of experiment to test theory

We developed a test construct which uses an unmanned system and military personnel to evaluate the application of suspicion theory to detection of cyberattacks in an operationally relevant scenario.

3. Execution of experiments, data collection, and analysis (next step)

We plan to conduct the experiments and collect / analyze the experimental data in early 2016.

4. Documenting results and observations (next step)

We plan to document results and observations from the experiment in mid-2016.

Literature Review and Application of Theory

After reviewing much literature in trust and suspicion, we determined the theory of suspicion, as proposed by Bobko, et al. [3], to be the most relevant literature to our study of operator response to cyberattacks due to its emphasis on perception of malintent (intent to cause harm) and focus on an information technology (IT) related context. This theory was developed under a research portfolio sponsored by the Air Force Research Lab (AFRL) / 711th Human Performance Wing (HPW) and funded by the Air Force Office of Scientific Research (AFOSR). We've been collaborating with Prof Bobko and AFRL to ensure the appropriate application of the suspicion theory and model to the study of operator response to cyberattacks. The theoretical definition of state suspicion proposed by Bobko, et al. [3] for an IT related context is:

"State suspicion is a person's simultaneous state of cognitive activity, uncertainty, and perceived malintent about underlying information that is being electronically generated, collated, sent, analyzed, or implemented by an external agent."

We provide a brief explanation of this theory from the point of view of the State-Suspicion Model (below, in Figure 16) developed by Bobko et al. [3] and conclude this section with a discussion of research propositions that we have taken from the literature review and applied to our study.

This model was initially developed and discussed in an information technology context.

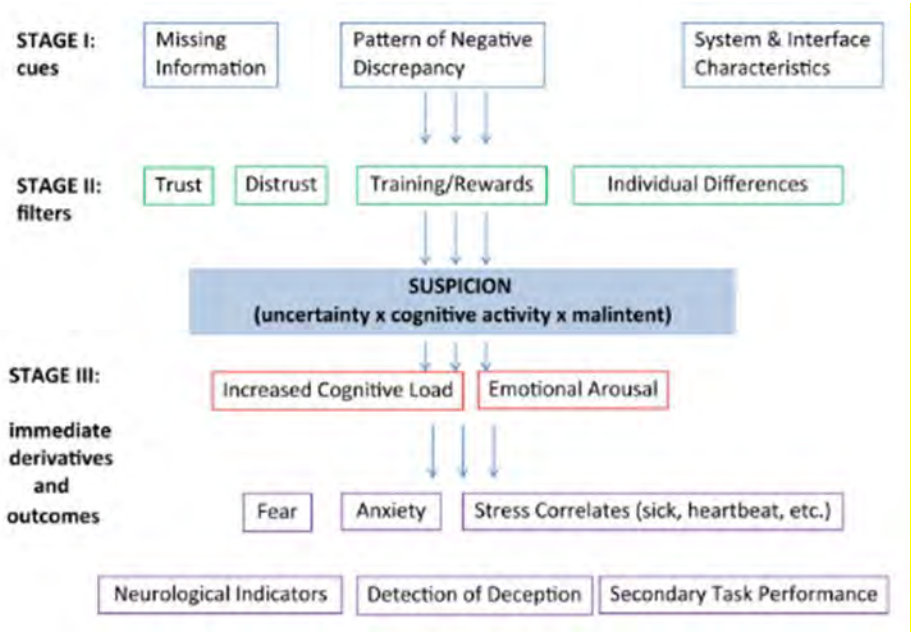


Figure 16: Stages of state-level IT suspicion

Stage 1 cues refer to indications from the environment which can act as a trigger of state-level suspicion. The boxes listed across the Stage 1 row are example categories of potential indicators and can serve as sources of manipulation for experimentation. A listing of more specific prompts is included in the article.

Stage 2 filters denote individual difference (trait-level) variables likely to affect state-level suspicion. In this model trust and distrust refer to an individual's propensity towards those factors. Schoorman et al. included a seven item measure for propensity to trust in their 2007 article and some researchers have reverse scored the measures to account for distrust [4]. Propensity to trust or distrust are interesting from a behavioral science view point, because they can both potentially affect suspicion. Suspicion is a cognitive process based in part on uncertainty, and both predisposition to trust and distrust remove some of that uncertainty [3]; since, by definition, those states are either certainty of positive or negative outcomes, respectively. Researchers believe trust may inhibit state-suspicion by deemphasizing Stage 1 environmental cues, and distrust may act as a catalyst for state-level suspicion [3], [5]. Since suspicious thought involves the cognitive generation and consideration of multiple plausible, rival hypotheses for observed behavior [3], [5], individual differences are also interesting to consider.

This is one reason scientists are concerned with the issue of task saturation or cognitive overload when designing a system. Essentially, the operational process needs to allow for a "reasonable"

baseline cognitive load. Therefore, a person who is creative and has "extra" cognitive capacity is believed to be more capable of engaging in suspicious thought while continuing normal operations [3]. Therefore, some of the trait-level attributes proposed to affect suspicion include a person's creativity, need for cognition, and cognitive capacity. These three factors along with propensity to trust are believed to create the trait-level factor capacity to become suspicious and serve as antecedents to state-suspicion [3].

Stage 3 of the state-suspicion model refers to potential outcomes (physical manifestations) of suspicion. Increased cognitive activity is the outcome of interest to my research. The other outcomes are important, but increased cognitive activity has known metrics such as NASA TLX and Instantaneous Self-Assessment making it a better measurement outcome. Researchers within the AFRL suspicion portfolio are working on measures for some of the other outcomes. For example, Professor Leanne Hirschfield of Syracuse University is working on measuring suspicion in the brain with functional near-infrared spectroscopy (fNIRS). Their research group has also developed a State-suspicion Index questionnaire, which we intend to use.

We have taken the following research propositions from our literature review:

- Suspicion leads to suspended judgement
  - Bobko et al. [3]; Hilton, Fein, and Miller [6]
- Trust inhibits suspicion; distrust can act as a catalyst for suspicion
  - Bobko et al. [3]; Lee and See [7]; McKnight et al. [8]; Buller and Burgoon [9]
- Suspicion lead to increased cognitive activity due to consideration of multiple rival hypotheses
  - Bobko et al. [3], Mayer and Mussweiler [5]
- General trait-level attributes and domain knowledge influences one's capacity to become suspicious
  - Bobko et al. [3], Mayer and Mussweiler [5]

We have developed an experimental process to link these theories to observable operator behaviors in an attempt to replicate, explain, and potentially predict operator response to cyberattacks as discussed in our design of experiments.

Design of Experiments

The primary interest of our experiments is to determine if suspicion improves operator detection of cyberattacks (a form of malintent). However, suspicion is a latent variable in our test design representing the hypothetical construct of suspicion. Theoretically, there are three components of suspicion: uncertainty, increased cognitive activity, and perception of malintent. All three must occur simultaneously for suspicion to occur; therefore, we must impute the occurrence of suspicion (and its effect) from the performance outcome measures. We've developed a test model which will enable us to conduct experiments and collect data to make meaningful observations. That test model is presented below. We will provide a brief explanation of the test model, and then give a test sequence flow for better understanding of the process.

Test Model

We will provide a short description of the test model in reference to the color coding.

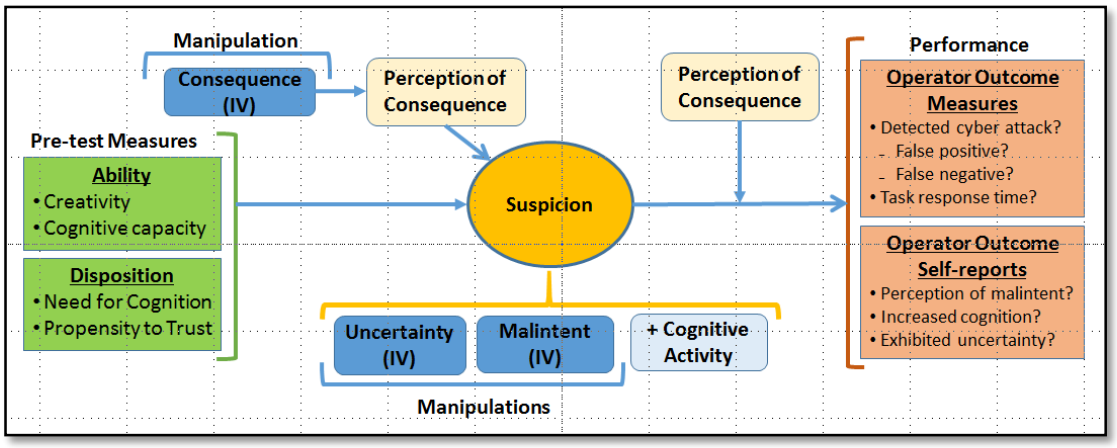


Figure 17: Test model for operator suspicion; one of the goals of Part I, Section 4 of SERC document.

Orange oval: The orange oval would be considered a latent variable representing the hypothetical construct of suspicion; therefore, we must impute the occurrence of suspicion (and its effect) from the performance outcome measures (salmon colored box on right).

Blue & Tan boxes: The model is a three factor x two level model, which means it contains three independent variables (IV) represented by the blue boxes, and each IV has two levels (i.e. Lo – Hi). The IV's of Uncertainty and Perception of Malintent represent two of the three major components of suspicion and will be manipulated. The third IV in the model is Consequence, and it will be manipulated. We are interested in the relationship between actual consequence (blue box) and operator perception of consequence (tan boxes) on suspicion (orange oval) and performance outcome measures (salmon colored box on right).

Light blue box: The third component of suspicion (cognitive activation) will be measured during the experiment and at the end.

Salmon colored box: Performance will be measured in two ways: Operator outcome measures and Operator outcome self-reports. The Operator Outcome Measures compare how the operator performed (e.g. detect, not detect, override Sentinel, etc.) against the actual sequence of events in the scenario (e.g. attack, no attack, Sentinel correct, Sentinel not correct, etc.). The Operator Outcome Self-reports are questionnaires to assess their perception of items such as uncertainty, malintent, cognitive activation, and suspicion.

Green box: The green box on the left represents pre-test measures we will assess for each test subject prior to their starting the experiment. These attributes are thought to be linked to one's capacity to become suspicious. We will compare the pre-test attributes with the post-test results to look for correlation between the operator traits and operator performance outcomes. The data will be collected via questionnaires.

Test Setup and Flow (Future Work)

We plan to conduct the experiments at AFIT using AFIT students as test subjects. The experiments will be supported by the Autonomy & Navigation Technology (ANT) Center at AFIT. We currently plan to utilize unmanned ground vehicle (UGV) systems as the representative cyber physical system. These UGV's use the same mission planning software and navigation components as the small unmanned air vehicle (UAV) systems operated by the ANT Center, and they routinely serve as test platforms (risk reduction) for hardware & software changes for the UAVs prior to implementation and flight test. The UGV system consists of a truck, ground control station (GCS) laptop with mission planning software, a wireless network, modem & radio controller (RC). The RC is for manual control of the UGV, and we do not plan to use manual control. The UGV will be equipped with a GPS, Pixhawk autopilot, RC receiver, modem, and power bus. Additionally, a Sentinel system is assumed to be part of the system and always operational. A picture of a UGV system provided below.

Figure 18: Unmanned Ground Vehicle (UGV) cyber physical system

The test subjects will be recruited as operators from the AFIT and AFRL populations. They will be randomly assigned to a minimum of three groups. Each operator will receive training specific to the experiment to normalize domain knowledge across the test subject population. Prior to starting a test sequence, each operator will receive pre-test trait-level questionnaires and measures will be taken regarding their individual attributes. The operators will then be asked to run through a series of eight mission scenarios using the system in autonomous mode (UGVs follow pre-programmed mission waypoints set by the operator). The order in which the operator participates in the scenarios is determine by the group to which randomly assigned. Each scenario will represent a different combination of consequence, uncertainty, and malintent manipulations along with differing exposures to cyberattack and Sentinel detections. The performance measures discussed in the model above will be collected for each test subject at the end of each scenario. In total, approximately 100 data points will be collected per operator for a test sequence (all eight scenarios).

Observations will be made and statistical analysis will be conducted on the resulting data set to assess the relationships of interest for this research effort.

Conclusions and Next Steps for Human Factors

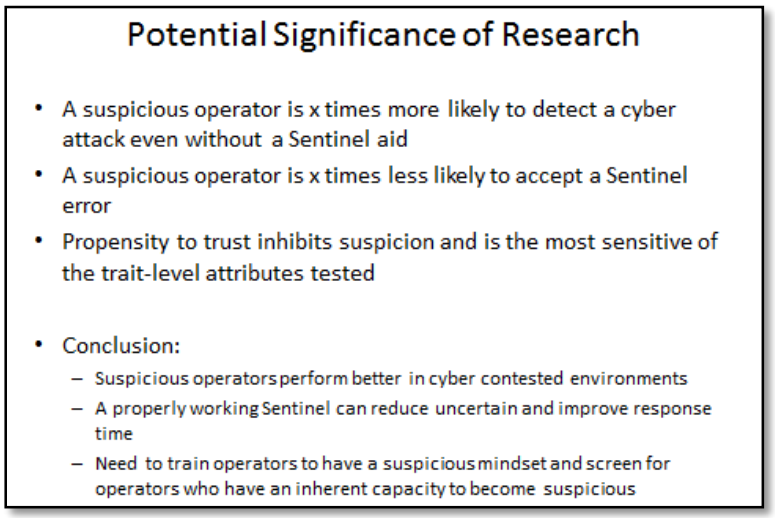
The 2013-14 UVA/MITRE project [2] noted that although there is considerable effort ongoing to prevent or detect and mitigate cyber-attack on DoD networks and IT systems, cyber-attacks on RPAS

has not been examined from an operational perspective. In order to bring the fight to the enemy, RPAS will need to effectively operate in a contested environment which includes the threat of cyber-attack. The Sentinel is a promising technology, and the capability it provides would most likely need to be designed into the system it protects.

These and other observations from the UVA/MITRE study provided motivation for our research into the human response to cyberattacks. We acknowledge the potential benefit of the Sentinel capabilities to serve as an aid to operators of cyber physical systems in the detection and response to cyberattacks; however, we note the Sentinel is potentially fallible and vulnerable to the very thing it is designed to protect against. Therefore, the operator must remain the ultimate decision maker even with the presence of an aid. Based on our literature review, we believe the concept of suspicion as proposed by Bobko et al. [3] (through AFRL & AFOSR sponsorship) has potential to develop a more resilient operator. We will apply this theory of suspicion to the study of operator response to cyberattack in the context of a cyber physical system. We will use an AFIT ANT Center UGV as our representative cyber physical system, and we will recruit AFIT and AFRL personnel as operators in our experiments. Table 2 (below) shows compliance with Part I, Section 4 of the SERC Research Topic document.

We have developed a design of experiment concept to rigorously test the theory of suspicion and its relationship to operator performance against cyberattacks. The next steps include development and approval of the detailed test protocols by the AFIT and UVA Institutional Review Boards, recruiting and training the test subjects, conducting the experiments, collecting and analyzing the data and making observations and recommendations from the results.

The slide below provides an example (hypothetical) of the potential significance of this research.



Develop plans for experiments and define the data collection needs	Covered in Section 2 of this *Human Factors* Section
Develop a Simulation Support Environment	Covered in Section 4 of this *Human Factors* Section

Obtain Approvals to conduct human experiments	This is a huge undertaking, and has been deferred until FY 16
Conduct the experiments and document the results	FY 16

Table 2 Goals and the Research Components that Meet the Stated Goal

List of References for this Section

[1] R. A. Jones and B. M. Horowitz, "A System-Aware Cyber Security Architecture," Syst. Eng., vol. 15, no. 2, pp. 225–240, 2012.

[2] C. A. Gay, B. Horowitz, N. Lau, K. Leach, M. Dinsmore, M. Lewis, C. Jella, D. Neal, and J. Rush, "RT-115 Project Report - Part 2, Human Factors Engineering and System-Aware Cybersecurity," 2015.

[3] P. Bobko, a. J. Barelka, and L. M. Hirshfield, "The Construct of State-Level Suspicion: A Model and Research Agenda for Automated and Information Technology (IT) Contexts," Hum. Factors J. Hum. Factors Ergon. Soc., vol. 56, no. 3, pp. 489–508, Jul. 2013.

[4] F. D. Schoorman, R. C. Mayer, and J. H. Davis, "An Integration Model of Organizational Trust: Past, Present, and Future," Acad. Manag. Rev., vol. 32, no. 2, pp. 344–354, 2007.

[5] J. Mayer and T. Mussweiler, "Suspicious spirits, flexible minds: When distrust enhances creativity.," J. Pers. Soc. Psychol., vol. 101, no. 6, pp. 1262–1277, 2011.

[6] J. Hilton, S. Fein, and D. Miller, "Suspicion and Dispositional Inference," Personal. Soc. Psychol. Bull., no. 19, pp. 501–512, 1993.

[7] J. D. Lee and K. a See, "Trust in automation: designing for appropriate reliance.," Hum. Factors, vol. 46, no. 1, pp. 50–80, 2004.

[8] D. Mcknight, V. Choudhury, and C. Kacmar, "The impact of initial customer trust on intentions to transact with web site: A trust building model," J. Strateg. Inf. Syst., vol. 11, pp. 297–323, 2002.

[9] D. Buller and J. Burgoon, "Interpersonal Deception Theory," Commun. Theory, vol. 3, pp. 203–242, 1996.

Developing Cloud-Based Sentinels and Integrity Monitoring

Another important component of this project (the 3rd in the list of 4 called out in Part I, Section 4 of the SERC Research Topic document) was conducted by Prof. Marty Humphrey and involved research into the practicality and usefulness of putting *sentinel* functionality into dual *cloud* architectures.

In CY2015, Professor Humphrey and his team continued their research and development into integrity assurance of private cloud infrastructures (as detailed in the SERC Research Topic document).

The initial thrust was to expand the types of attacks that can be recognized and defended against. Active testing was implemented and expanded from a probe concept from the previous year to include the development of and experimentation on a diverse set of integrity assurance functions and to operate those protections while the cloud is performing its operational sentinel functions. More specifically, this year the team focused on attacks aimed at changing important private cloud data (reading confidential cloud data, modifying cloud data, modifying DevOps scripts, disabling/disrupting internal cloud monitoring routines, etc.) In addition, the team expanded the attack surface beyond the virtualization layer (HW) and into the virtual machines themselves. The team redesigned the mechanisms to be easier to plug-and-play new protection mechanisms and policies. The monitoring system was integrated into the *OpenStack* administrator's console to make viewing of potential problems a quick and easy task.

An additional thrust was to integrate with the physical sentinel research, which was a separate activity in the broader research project. The research team obtained a prototype of an AIMES-like cybersecurity sentinel ("Piccolo"), running on an *OpenStack* deployment. The team spent significant time understanding how the sentinel system interacts with the cloud, in order to focus on protecting the aspects of the cloud that were most important to ensuring the correct operation of the sentinel system.

The demonstration at the Sept 2015 site visit focused on an attacker covertly discovering and modifying a key sentinel configuration file, with the intent of (at least) a *Denial of Service* attack. The active probe infrastructure recognized this modification. The system response was two-fold: first, the infrastructure was dynamically switched to an entirely separate cloud (in this case, the public Cloud Azure, as a proof of concept), continuing to successfully and securely running the sentinel. The second aspect of the response was to not immediately shut down the existing cloud, but rather retain the running infrastructure to ensure the ability of diagnostics and forensics to assess the scope of the damage.

To further satisfy the goals of Part 1, Section 4 of the SERC Research Topic document, additional research and development expanded this effort to different classes and configurations of private cloud infrastructures. Previously, the team had operated the deployment and test cloud in a very specific configuration on a very specific and limited computing platform. The key aspects of this part of the project are:

- ***Non-virtualized test environment***. A virtualized environment provided the best platform by which to design and implement (and test) modifications to the OpenStack software to measure and ensure cloud integrity. In this year, the architecture and modifications were tested on non-virtualized hardware obtained through the RackSpace public Open Cloud (the

"metal" servers). The team confirmed that the performance expectations (e.g., of the timing/duration of certain cloud operations) from the virtualized test environment were valid in the non-virtualized environment. For example, the team's monitoring infrastructure aimed at measuring the duration of "typical" cloud operations were confirmed to be valid and operational in the non-virtualized environment.

- ***Different versions of OpenStack***. A new version of OpenStack was released in mid-project-year. The team adapted and modified their existing framework and mechanisms to work in the new versions of Openstack (from "Icehouse" to "Juno" and "Kilo").
- ***Different configurations of OpenStack***. OpenStack provides the flexibility for a wide set of configurations/redundancy of key Cloud services. To ensure that the designed framework and mechanisms operated correctly in multiple OpenStack configurations/deployments, the team deployed 4 different system architectures of OpenStack using "metal" servers on the RackSpace public Open Cloud and confirmed the valid operation of the team's active probing infrastructure.
- ***Geographically diverse clouds***. The team's tests were expanded to include wide-area and multi-region OpenStack clouds by utilizing the geographically diverse capabilities of Amazon Web Services and the RackSpace public Open Cloud. These tests accurately simulated a deployed geographically diverse OpenStack deployment, and the results confirmed the team's active probing mechanisms work in such a diverse environment.
- ***Multi-clouds.*** Multiple, disparate OpenStack clouds were created, whereby one cloud was monitored and the second cloud was used in the event of compromise of the first cloud. An application-specific design approach was pursued to facilitate the "roll-over" to the second cloud (i.e., general-purpose mechanisms to keep two clouds "in sync", irrespective of the applications on the cloud, is impractical). Upon successful tests, the team pursued (and completed) using the Microsoft Azure public cloud as the "roll-over" cloud. This capability was successfully demonstrated in the Sept 2015 site visit.

See Table 3 (below) for compliance notes for this 3rd of 4 topics:

Further develop the AIMES cybersecurity sentinel	The Piccolo cybersecurity sentinel provided a simplified approximation to AIMES; Piccolo was both modified/redesigned to operate as a cloud application and was instrumented to provide the foundation for itself to be monitored for accuracy/correctness. The lessons learned from the team's use and modifications to Piccolo provide the basis for the cloud-based use of AIMES.
Develop and conduct experiments on a diverse set of assurance functions	This year the team focused on attacks aimed at changing important private cloud data (reading confidential cloud data, modifying cloud data, modifying DevOps scripts, disabling/disrupting internal cloud monitoring routines, etc.). The team's mechanisms were validated across a wide range of cloud configurations, including virtualized OpenStack deployment, non-virtualized OpenStack deployment, different versions/configurations of geographically-contained OpenStack deployment, and single geographically-diverse OpenStack

	deployments. Roll-over multi-cloud tests were conducted involving OpenStack, Amazon Web Services, and Microsoft Azure.

Table 3 Compliance with Part I, Section 4 of the SERC Research Topic Document

Expansion of the System by Moving Components Outdoors

With the milder Fall weather, the team repositioned the 'building under protection' and (therefore) the sensor pods to the configuration shown in Figure 19 (below). In this manner, some of the sensors (including the UAV) can be positioned OUTDOORS, greatly alleviating or perhaps eliminating the aforementioned GPS problems. The UAV can hover outdoors, where it can pick up GPS signals, be unconstrained by the building (with operator care to avoid the trees), and can hover out of range of the pod's audio and motion sensors—making the resulting system more in line with the original vision shown in figures 1-6. The acquisition of surplus Unattended Ground Sensors (UGS) from either the DIA (Defense Intelligence Agency) or NGIC (National Ground Intelligence Center) would expand the sensor options and greatly enhance the total system capabilities. Both of these agencies are nearby, just to the north of Charlottesville.

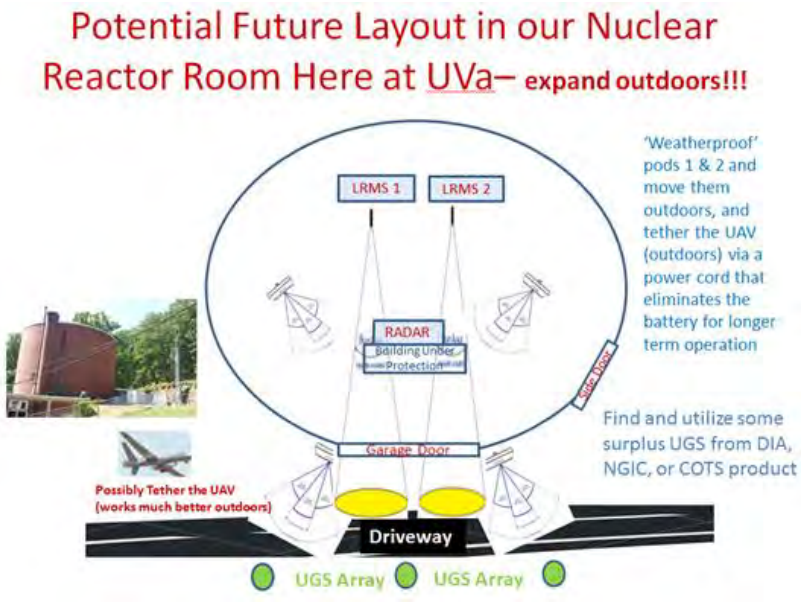


Figure 19: Moving the UAV and Two of the Four Pods Outdoors

Other Additional/Miscellaneous Capabilities and Monitoring

NMAP Network Monitoring

One additional capability for monitoring is easily added with network scanning programs like NMAP or Nessus. In an unmonitored environment, it is fairly easy for an adversary to install a device (physically, or wirelessly) on a network, monitor the traffic for a while, and then devise some methods of trying to 'spoof' other devices on that same network and inject confusing traffic, counter command & control signals, listen for sensitive communication, or actually engage in *Denial of Service* (DoS) activities to really confuse the normal operations being performed on the network. One way to counter this IS to do some active monitoring of just who (by IP and even Ethernet address) is ON your network, and try to sort out the legitimate devices from any newly installed fraudsters. NMAP makes this particularly easy by storing previous network scans (where it reports on the open ports of each

and every device on the network, by IP address) and then allows you to compare a recent scan with a 'baseline' (previous) scan to see if there are different IP (or Ethernet/machine) addresses on the network that were not there before. This is a useful way to keep track of the devices on a network, and to make sure no one has added (or SUBTRACTED) anything. Figure 20 (below) shows an NMAP scan of the 192.168.30.0/24 network, using ZenMap as the front end GUI. Again, comparing daily scans against the previous day's scan (perhaps as much as a few times a day) would give more confidence that intruders had not modified the network in any detectable way.

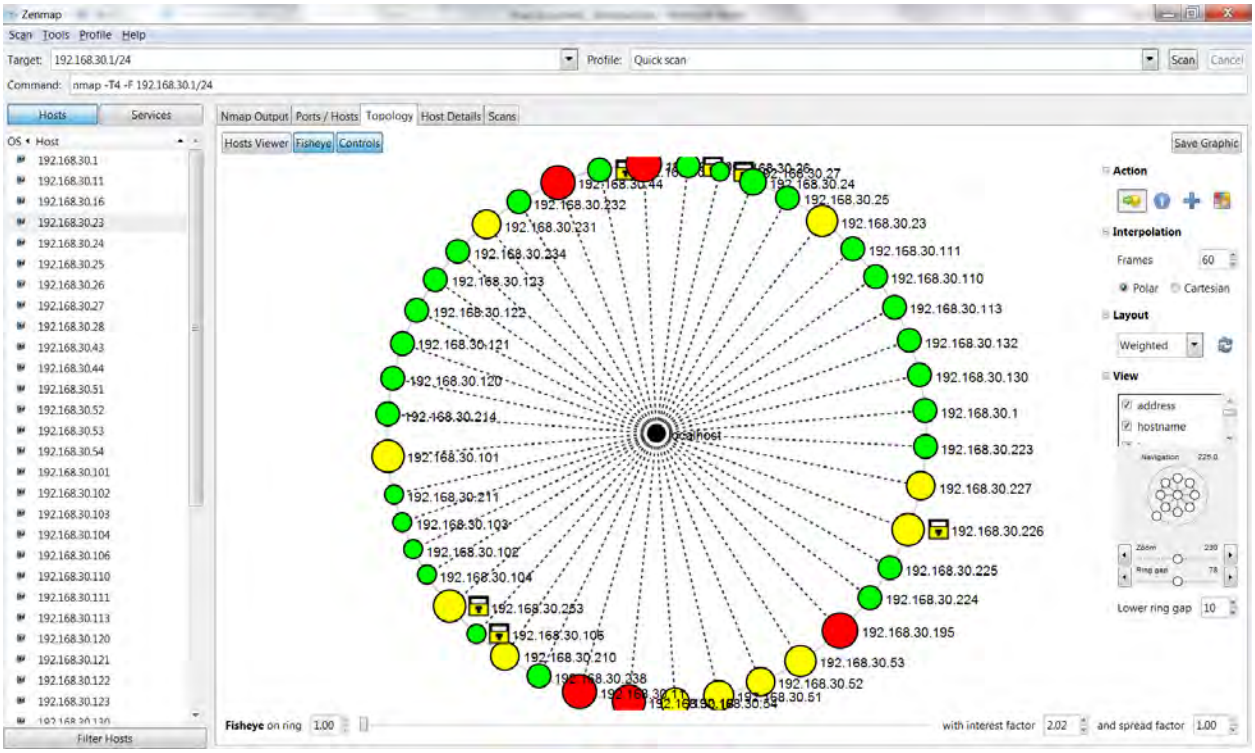


Figure 20 An NMAP Scan (Using ZenMap as the Front End GUI) on our 192.168.30.1 Network

Other SIEM tools like Splunk, Solarwinds, Tableau, etc., useful for monitoring logs of firewalls, IDS, and antivirus software packages can assist in watching for suspicious activity on sensitive networks. These automated SIEM tools do a lot to minimize the workload that had to be manually expended in the past to do such active monitoring of live networks.

Architectural Selection and Assessment Methodology (MissionAware 1.0)

Executive Summary For This Section

This section of the report, supplied by Professor Peter Beling and his team, describes the MissionAware Architectural Selection and Assessment methodology for advanced system modeling and attack trees. The objective of this methodology is to: 1) identify the critical system components for a particular mission, 2) to derive the possible attack paths to attack those components, 3) to determine which of those attack paths would be most desirable to an adversary, 4) to identify possible cyber security defenses against those attacks as well as to evaluate the impacts of those defenses on the attacker, 5) to assess the effects on system performance of potential defenses, and to 6) estimate the security trade-offs of the various architectural solutions. The MissionAware framework builds off the SysAware methodology developed in 2013 and 2015 project phases, and satisfies the goals of the 4th item in Part I, Section 4 of the SERC Research Topic document.

The relational SysAware methodology is composed of these six steps; each step having a well-defined goal, required deliverables, and responsible team(s) for that stage. The first SysAware methodology, SysAware V1.0, focused on a manual approach to implementing these steps. The manual processes included (i) a scoring system that highlights cost-benefit tradeoffs for decision makers and (ii) a method based on influence diagrams designed to aid decision makers in understanding ways in which uncertainty could be introduced into an attacker's outcomes. The SysAware V1.0 methodology was successfully applied to the UAV surveillance system, both to develop the concepts through case studies and as a practical method for selection of research targets for the attack and defense teams. From these applications, it became apparent that while the methods left participants with a clear sense of having successfully explored the design space, the manual effort they required would not scale well with system size and complexity.

As another interim step, SysAware V2.0 represented further progress toward automating the processes for evaluation of the architectural selections of critical system functions to be protected on a cyber-physical system. The SysAware V2.0 approach is based on exploiting modeling paradigms and open source or commercial software. Specifically, the System-Aware Architectural Selection and Assessment methodology is recast in terms of use of model-based systems engineering tools, such as SysML, and attack tree tools, such as the commercial package SecurItree. Much of the project effort was directed at developing concepts and software to integrate systems models with attack trees to facilitate an iterative architectural design process in which decision makers could easily switch back and forth between defender and attacker views of the system or mission. The Phase 2 activity culminated in a workshop in November 2014 with participants from 10th Fleet Cyber Command and the Johns Hopkins Applied Physics Lab. The goal of the workshop was for participants to discuss the complexities of the decisions and tradeoffs inherent in choosing a defensive architecture as well as to introduce a methodology and toolset being designed to support decision makers.

A principal conclusion of the workshop was that future development of the methodology should focus on a mission rather than systems perspective, and also should include more automation and productivity enhancement tools to mitigate the complexities of systems modeling. MissionAware introduces new methodological elements that address these issues. By broadening the focus of the user experience to include more stakeholders beyond the system modeler and reasoning with existing knowledge of a component and attack library, the MissionAware War Room increases the productivity

of model-based acquisition. These increases, realized over time, are enabled through capture of a mission's required flow of information with a consistent modeling approach and continuous assessment of the attack surface of a mission configuration. These features are in turn enabled by the SystemAware SysML profile, providing a baseline for description of Cyber-Physical Systems (CPS), the Composable Attack Pattern for capturing the otherwise unmodeled or unmodelable weakness of a mission, and the Evolutionary Assessment Tool (EAT), providing immediate insights into the evolution of secure mission architectures.

Section Introduction (MissionAware; Advanced Systems Modeling & Attack Trees)

At present, the state of the practice for augmenting Cyber Physical Systems security is primarily perimeter based security (such as firewalls, intrusion detection mechanisms, anti-viral signature software, encryption, and advanced user authentication). These methods are termed *tactical cyber methods*, because they are mainly static solutions for known attack patterns. However, the trend in adversarial attacks is moving toward well-formed coordinated multi-vector attacks that compromise the system in such a way that detection and identification of the attack is challenging for perimeter security solutions and human monitoring alone. These style of attacks are possible due to the recent advent of stealthy Advanced Persistent Attacks (ATP's) - where attackers penetrate firewalls and intrusion network devices by very sophisticated malware, and then quickly steal information on the operational aspects of critical infrastructure systems before they are detected, if they are detected at all ([1]). The information retrieved by an adversary from a "smash and grab" ATP attack is subsequently used to craft special multi-vector malware that targets the critical digital components of infrastructure systems.

As an end result, it has been recognized that perimeter security needs to be strongly augmented by other approaches for addressing the current and emerging cyber threat potential for Cyber Physical Systems (CPS) ([2], [3], [4]). Reliance and dependability of CPS will depend upon not only on the understanding of threats, but also developing well-engineered science based approaches to protect critical assets that; (1) are effective at deflecting multi-vector APT attacks, (2) do not interfere with safety, dependability and reliability aspects of the CPS, and (3) reverse the cyber asymmetry from the attacker advantage to the defender's advantage.

Vulnerability analysis today is performed largely at two extremes as shown in Figure 21. At the very low level which is typified by binary analysis and debugging and packet analysis tools, and conversely at the very high level, which includes tools like attack trees, graphical methods, and tabular auditing methods. At the low level, the focus is on very specific interactions at the program machine interface level which largely excludes larger system context. At the higher level, the focus is on system level with focus on postulated attacks or vulnerabilities with respect to the system, with little support to determine what is the real set of vulnerabilities and possible exploits of those vulnerabilities. Consequently, today we practice cyber-systems vulnerability analysis in a disjointed, fragmented fashion that leaves exposures to exploits undetected due to the fact that a systems perspective of cyber-vulnerability analysis is lacking. We argue that a model-based systems perspective is a potential solution to this dilemma, guided by well-formed principles of architectural level cyber analysis this team investigated in Phase 1. This brings the power of system level modeling to enable a holistic perspective on cyber security.

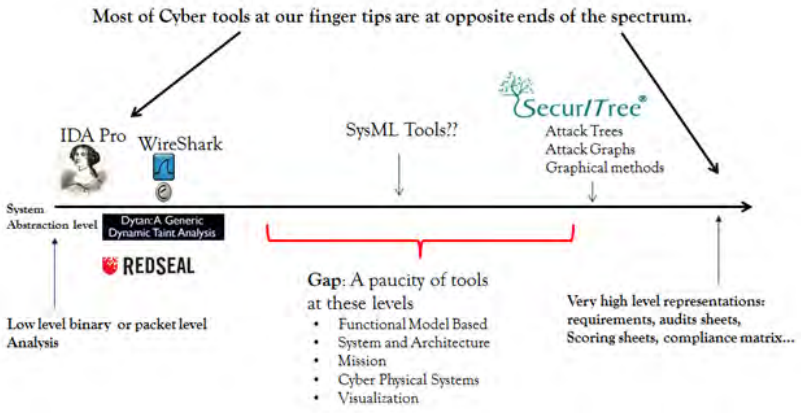


Figure 21: Spectrum of approaches to vulnerability analysis.

The key advantage of employing a model based Cyber-Vulnerability Analysis (CVA) approach is viewing the system from several important perspectives and dimensions of abstraction within a framework. These perspectives include the *mission perspective* which provides context to the consequences of exploiting vulnerabilities, the *architecture domain* which embodies the organization and relationships between subsystems and platforms, and the *platform domain* which represents all of the functions and components that are working together to achieve a mission. Figure 22, shows the concept of model based CVA:

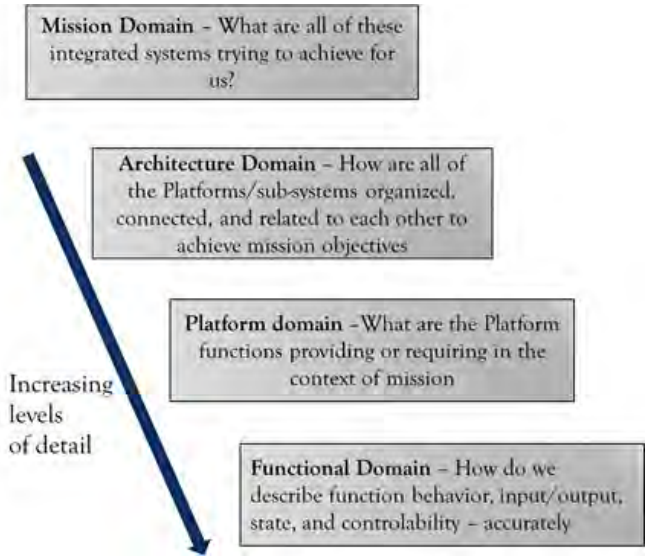


Figure 22: Levels of cyber vulnerability analysis.

Model based analysis allows us to support decision making by providing reasoning along these dimensions. It provides a model to collect insight that otherwise could be overlooked. It allows us to integrate information from vulnerability analysis tools across the spectrum of Cyber Vulnerability Analysis (e.g., attack trees) to the framework. It enables us to assess the criticality of platforms and functions with respect to mission, and finally it allows us to evaluate potential cyber-defenses in a fair and objective manner for classes of attacks and threat agents.

Another important question is, “why do we want a multi-perspective model of the system?” While modeling at this level requires a modest upfront investment, the model becomes a living

representation of the system, allowing system engineers and administrators to regressively evaluate the impact of modifications or upgrades to the system in the context of preserving security for various missions. Most importantly, the information extracted from the model (via querying analysis) allows decision makers to have higher levels of confidence in the cyber security solutions. The model based approach has the potential to provide insight into important questions such as:

- Did we overlook important attacks or important defenses?
- How do we know if an attack/defense is important?
- If we do choose to add a defense solution, in what way does this change the system and how should it impact our perception of vulnerability?

MissionAware Architectural Selection Framework

A key objective in Systems-aware Cybersecurity is to reverse the asymmetric advantage enjoyed by attackers by identifying areas where the defensive team can make minimal changes to the system that will cause a maximum increase in difficulty, uncertainty, or expense for the attacker. A wide variety of design patterns for functional defense have been developed using concepts such as data provenance, moving targets diverse redundancy, voting schemes, and sentinels. Systems-aware Cybersecurity solutions have been developed for a number of cyber-physical systems including shipboard navigation systems, power-generating wind turbines, and video reconnaissance systems on unmanned aerial vehicles (UAVs).

The most basic decision problem in designing a systems-aware cybersecurity solution, which we term the architectural selection problem, is that of selecting which system functions to protect and then choosing a collection of defensive measures from among a rich library of design patterns. To make such choices, the designers of defenses should have deep understanding of the system to be protected, including an understanding of the relative importance of each function in the estimation of the system owners

Finding the best solution is a complex multi-criteria decision analysis problem, with several distinguishing characteristics:

- Solutions must account for the actions and capabilities of the adversary, both now and in the future.
- Solutions must satisfy a large and diverse group of mission stakeholders.
- A mission solution involves multiple systems each needing to be secured.
- Solutions for a mission area must account for the particulars of each of the systems being protected and their relationships to mission outcomes.
- There is need for a decision making process with supporting tools that will help stakeholders select a satisfying solution for each mission area

Architectural Selection Methodology

The SysAware cyber assessment methodology described here was designed to be an iterative process that relies on inputs from a range of stakeholder communities. In order to ensure that the information being used is as accurate and certain as possible, it was imperative to ask individuals questions that were appropriate to their backgrounds and areas of expertise. This is accomplished by initially dividing the stakeholders into three distinct groups:

- **Red Team** - The red team is made up of individuals with knowledge of cyber-attacks and potential threat agent classes. Their work is focused on developing candidate attack vectors and assessing the effectiveness of the proposed design patterns.
- **Blue Team** - The blue team consists of designers and users of the system being protected. Their responsibilities include identifying and prioritizing the critical system functions to protect, as well as determining which security design patterns can be implemented on which system functions.
- **Green Team** - The green team, which is comprised of experts in system cost analysis and adversary capability, analyzes costs, to both the attacker and defender, for candidate architectural solutions.

The iterative steps of the architectural selection methodology are as follows:

Step 1: Define the Variables and Relationships within the System to be Protected

The initial step of the methodology is focused on framing the problem to ensure that all participants in the process are on the same page regarding the system to be protected. The process begins by identifying the critical functions of the system and defining the variables and influence relationships within that portion. Step one is to be performed by the blue team and is intended to outline the expected functionality of the system with minimal defensive strategies implemented. At this point, a system influence relational diagram is constructed using directed acyclic graph (DAG) notation. This diagram is created for the system without the consideration of a cyber-attack to ensure that everyone involved in the process is in agreement on the most basic structure and components before the 'complication' of an adversary is added.

Step 2: Identify the Possible Paths an Attacker Could Take to Exploit the System

Step two introduces one of the issues that make this specific problem unique: an intelligent adversary. While the system influence relational diagram represents a system where success may be compromised by random failures, the cyber security architecture selection problem introduces concerns where the decisions made by an active player in the system can also compromise mission success. In step two, the red team is tasked with constructing an attack tree for the system functions identified in step one. By looking at the system from the perspective of an adversary, attack trees can be utilized to understand the possible paths an attacker could take to exploit a specific feature of the system.

Step 3: Determine the Subset of Attack Actions Most Desirable to an Attacker

Considerable analysis can be conducted after the construction of an attack tree. However, rather than focusing on quantitatively calculating the probability of success for a specific attack path as is typically done in attack tree analysis, the analysis included in this framework considers a more qualitative and abstract metric space. In step three, the green team develops a set of variables that can be used to assess the difficulty of a particular attack path. These variables are called behavioral indicators and can include, but are certainly not limited to resources such as: technical ability, time, manpower, money, equipment, facilities, presence of an insider, and access to system design information. These variables are used to make two separate types of judgments: leaf node assessments and adversary profile construction.

Step 4: Identify Appropriate Defensive Actions and Their Impacts on the Attacker

After the red and green teams have identified the actions that an adversary would need to take to successfully execute an attack and the subset of those that are most attractive to a particular adversary, the blue team can then determine which of their existing defensive actions may be appropriate. The relational methodology relies on the assumption that a portfolio of design patterns has already been developed—either by previous blue teams or by an external group no longer involved in the process. If the current blue team was not responsible for developing the set of design patterns, it is assumed that they have access to the portfolio and the have the necessary knowledge regarding the meaning of each design pattern.

The goal of step four is to select design patterns from the existing portfolio that could be implemented to make the actions captured in the leaf nodes of the attack tree less desirable to the attacker. This can mean increasing the difficulty, cost, or probability of detection to the adversary or lessening the consequences felt by the defense in the case of a successful attack.

Step 5: Evaluate the Impacts of the Selected Potential Actions on the Defense

While step four captures the design patterns' impacts on the adversary, step five transitions to evaluating how those same choices impact on the performance of the system to be. The green team is able to apply their second class of intelligence information here; namely, cost analysis estimates for the defensive solution choices. At this point, each of the design patterns selected in step four is evaluated in regards to implementation cost, life cycle cost, and collateral system impacts. The green team is responsible for estimating the monetary cost of a solution, but the blue team also adds input on a solution's collateral system impact here. The blue team performs the evaluation of the solution's collateral impacts since they have knowledge regarding the system, how it will be used, and what impacts are unacceptable. Any solutions that are deemed to be beyond the allocated budget for System-Aware security (or introduce unacceptable impacts on system performance) can be eliminated from further analysis at this point.

There is one deliverable for this step: a reduced list of possible defensive choices, filtered from the original existing design pattern portfolio, to only those that increase the difficulty for the considered attacker while still remaining at an acceptable impact to the defense.

Step 6: Weigh the Security Trade-offs to Determine Which Architectural Solutions Best Reverse the Asymmetry of a Potential Attack

The goal of the sixth and final step is for all three teams to participate in a collaborative discussion regarding the security trade-offs that exist with the potential choices determined in step five. While each defensive strategy remaining after step five provides some potential security benefit, has an acceptable impact on the system being protected, and fits within the allocated budget the exact mixture of security to defense to budget varies by solution.

Modeling System Components: SysAware 2.0

The MissionAware methodology makes use of the SysAware 2.0 framework for modeling system components.

Model Based Engineering Approach

SysAware 2.0 exploits the utility and power of model based engineering methods to support CVA. Two structured modeling approaches, *SysML Models* and *Attack Trees*, are employed.

SysML System Models using model-based systems engineering tools and languages such as SysML. SysML provides a mechanism for the detailed description of the system structure, functions, and information flows in a searchable data format. The models allow one to capture the relationships between functional system entities and to recognize patterns (data, dependence, control) within the system. They also facilitate representation of the system attack surface to mitigate the danger of under modeling system attacks. SysML models can be used to represent the initial system "as-is" with minimal defenses and again with possible security solutions implemented. This perspective can demonstrate the value of solutions in the context of the whole system and an understanding of the complexity added to an attack by particular defenses.

The highest level concept in our model framework is the SysML block *Mission Context*, representing the highest level of analysis at a given time. The high level components of the mission, such as platforms, operators, and sensor assets are modeled as *parts* of the *Mission Context*. When an attack or defense is introduced to the model, it is added via *specialization* of the original mission, with a number of custom modifications possible beyond the feature-set provided by the SysML specification.

To model this mission-centric perspective, we chose to first decompose the mission-of-interest's system-of-systems in terms of requirements, structure, and a limited sense of behavior. Within those broad categories, we predominantly employed the concepts of requirements, use cases, blocks, ports and flows, generalization/specialization, and activities. This "minimalist" approach to modeling has a specific purpose - balancing simplicity, complexity and comprehension. Modeling is the art of describing "just enough behavior" to provide effective predictive behavior - capturing 1st and 2nd order effects. For Cyber Vulnerability Analysis (CVA) we assert that the current trend of developing detailed fine grain behavior models is not universally necessary.

Attack Trees are used to identify possible paths an attacker could take to exploit the system. These models use assessments of the attack actions and the attackers' capabilities to determine the subset of most preferable actions. By incorporating factors/attributes that characterize the adversary into the attack tree analysis, judgments can be made about which attack actions may be more desirable to the adversary. However, like most risk analysis methods, Attack Trees rely on Subject Matter Experts (SME) to convey opinions on threat agent capabilities/resources, the target system, and to simultaneously relate this knowledge with respect to the target system can impart uncertainty in the analysis process. Another issue with most attack tree formalisms (but not all) is that they are largely static. Meaning they cannot take into account dynamic behaviors such as; (1) threat actor actions based on state of the system, (2) phased attacks over time, or (3) sequence dependencies within steps in an attack.

Synergism Between Attack Trees and SysML

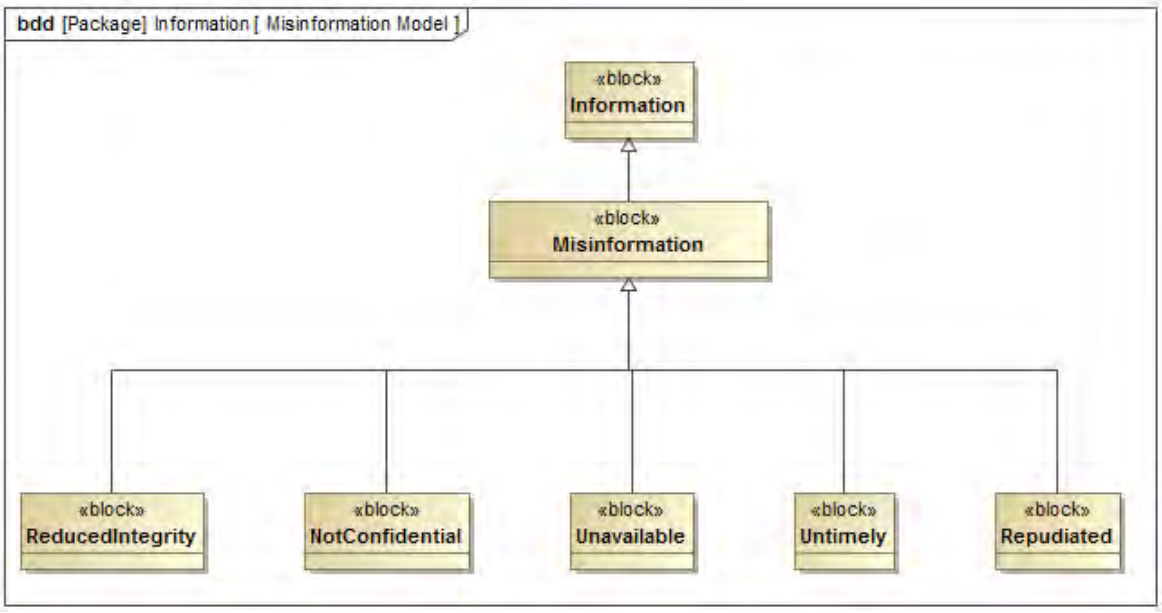
In the 2014 effort, we investigated ways that system level SysML models could support attack tree analysis to significantly improve vulnerability accuracy and completeness. We call this approach *Model Based Cyber Vulnerability Assessment*, which is a derivative of Model Based Systems Engineering paradigm. Part of this effort focused on what aspects of model based engineering can inform attack-tree security models to improve consistency of models, realism of attack patterns and discovery of vulnerabilities. The other aspect of this effort was to develop a basis for integrating attack tree

disciplines and system modeling processes in an effort to explore a more holistic view of the system - to capture greater insight into vulnerabilities, unforeseen relationships between system functions, navigating the decision space, and how to craft defenses that are cost effective.

Components: Inheritance, Structure, and Attributes

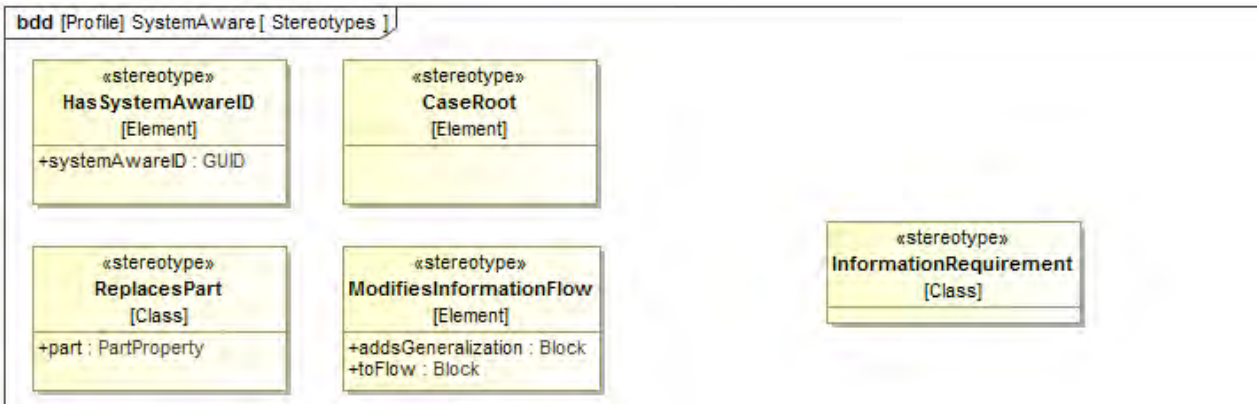
A model-based systems engineering approach adds levels of abstraction to the understanding of the system in a way that allows other users to more easily understand its structure, behavior and connectivity.

Wherever possible, when extending SysML to address the needs of the MissionAware approach, we use the language constructs of the SysML specification. For example, Information is a Block type, which is in turn specialized into different types of conveyed information, most importantly “Misinformation”, as shown below:



More generally, the specialization/generalization paradigm is crucial to selecting the appropriate level of modeling in a mission configuration. For example, an attack pattern which targets an un-branded router is at a lower level of specificity than an attack pattern which targets a Cisco 1841 router.

When the semantics of SysML are insufficient for a particular level of metadata, stereotypes gathered in the SysAware profile are used to denote the language modifications required.



With these added levels of information, each SysML model can be unambiguously interpreted as the "pattern" for fragments of a system configuration: the collection of instances which can be observed to meet a requirement, possess an attack surface, or embody some other system-level metric.

Cyber-Security Modeling - Susceptibility, Accessibility and Capability

A military operation is usually comprised of individual soldiers, mission specialists, forward operating command centers supported by a diverse set of systems like ground vehicles, airborne platforms, blue force trackers, unmanned ground and aerial vehicles, wireless communication networks, and unattended sensor networks – collectively working together toward a mission objective. The distinguishing characteristic of modern military operations are their substantial reliance on cyber systems and data to enhance sharing, precision, integration, interoperability, and decision making. From a systems viewpoint, modern military based systems are best categorized as "A System of Systems" - systems that cooperatively share their resources and capabilities together to form a new system which offers more functionality and performance than simply the sum of the constituent systems (wiki definition). As system complexity and our reliance on these systems increase, the consequences of a successful cyber attack on a given system or subsystem becomes more critical to overall mission planning and success. The complexity of "System of Systems" scale systems can strain traditional policy based and intrusion detection cyber security methods. As part of our phase II effort, we recognized the need to evaluate effective and efficient methods that could tackle large "attack surfaces" of system of systems. Attackers can probe and discover weaknesses of systems at their pace, and exploit system features including weak access points, undocumented features, privileged processes, and unintended system functionality to achieve their goals. We contend that the first steps of managing complex "system of systems" is a model based on the following corresponding elements - which we assert are necessary and sufficient for the presence of a successful attack.

Susceptibility - an inherent system weakness that can allow data, program actions, inputs, or outputs to be compromised through the loss of integrity, confidentiality, availability leading to an impact on mission prosecution.

Accessibility - The ability of an unauthorized agent to access the system and exploit one of the susceptible features of the system. Accessibility characterizes the direct and indirect channels or means for ingress into the system.

Capability - Characterizes the lower bound on the technical capability, skill, resources required to execute a given attack with respect to a given threat agent.

We further assert that only when these three threat elements are present does an actual vulnerability exist. This synergy relationship is show in Figure 23 below.

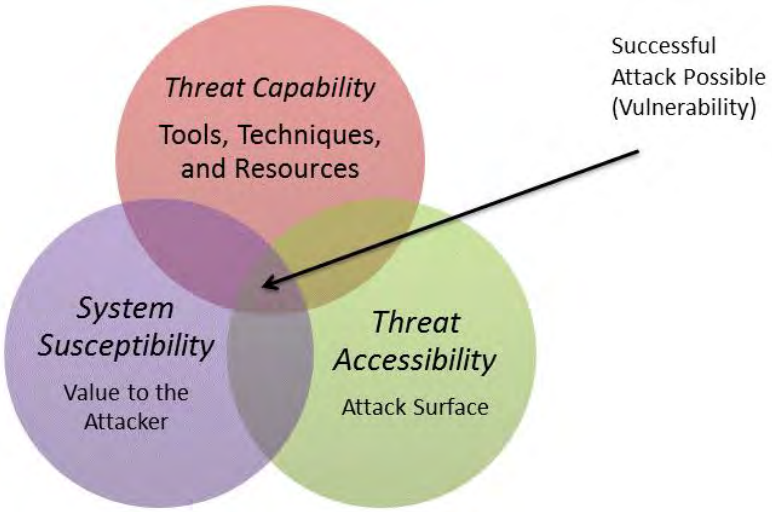


Figure 23: Representation of synergies needed for successful attacks.

Formal Definition of Attack Surface and Attack Patterns

An attack surface is roughly the set of system's actions resources which can be modified via actions (attacks). The more extensive the attack surface is, the more vulnerable the system can be. Prerequisite attributes for an attack surface to exist are:

- Susceptibility - Is the system attackable, critical functions that lead to significant damage.
- Accessibility - Can the system be reached, what are the channels or means for Ingress

An attack surface of system Q is the tuple $w = (X, L_c, I, M, O, Ch)$

Where:

L_c *is defined as set of all components*.

In practical terms, L_c is a predefined library of components for a given mission as instantiated in a SysML model of system Q.

Let w be:

- X is a subset of components (resources) taken from L_c that comprises system Q. X_j is an instance of a component from L_c that is a resource in system Q.
- I is the complete set of inflows (entry points) for all of X. I_j is specific inflow for a component X_j.
- O is the complete set of outflows (exist points) for all of X. O_i is a specific outflow for a component X_j.
- M is function that maps X, I, *and* O to *images* of X, I and O.

$$M: X \underset{i}{\rightarrow} \widetilde{X}, \quad M: I \underset{i}{\rightarrow} \widetilde{I}, \quad M: O \underset{i}{\rightarrow} \tilde{O}$$

M defines the transformation of X, I and O in terms of integrity, confidentiality, timeliness, and availability properties. These are considered the "secure" states for *Q*.

- *Ch* is a function that assigns to each entry or exit point of a component X_j an attack pattern *A* for all of *X and* $\tilde{X}$ in *Q*

$$Ch: I_j \cup O_i \rightarrow X$$

$$Chs: \tilde{I}_j \cup \tilde{O}_j \rightarrow \tilde{X}$$

Intuitively, a system's attack surface is the subset of the system's resources (methods, channels, and data) used in attacks on the system. The Attack Surface describes all of the different points where an attacker could get into a system, where they could get data out, or how they can affect the flow of information through the system (path connection between components) or finally how they could alter outputs. Not all resources, however, are part of the attack surface and not all resources contribute equally to the attack surface measurement. The grand challenge for complex systems is that the attack surface is not static – it changes with mission, threat capabilities, attack patterns, system upgrades, and human interactions. One of the prime reasons we chose to explore intelligent graph searching methods for this phase of work is based on the complex dynamic nature of the attack surface of real systems. Our goal for this phase of work is formulate a credible characterization of the attack surface from a quantitative perspective - to determine the contribution of each resource (*X,O, I*) to the overall system's loss or breach of security low level security properties e.g. *M*(*x*). With this characterization, we can semi-automate the search for vulnerabilities in the system attack surface.

In order to detect a system's attack surface, we need to identify the relevant resources that are part of the system's attack surface with respect to a set of mission requirements and baseline attack patterns. A resource in the system is part of the attack surface iff (if and only if) an attacker can reach the attack surface and use the resource in attacks on the system. In the above definition of attack surface we introduce the notion of entry points, exit, and components to describe these relevant resources. We assert any given resource's contribution to the attack surface reflects several important indicators such as; (1) the likelihood of the resource being used in attacks, (2) the criticality of the resource to the attack progress, and (3) the effort for an attack to make progress. For example, a task running with super user privilege is more appealing to an attacker and thus more likely to be used in attacks than a method running with user privilege.

From a practical viewpoint, the attack surface of a task or sub-system can be defined in terms of everyday use:

1. the sum of all paths for data/commands into and out of the application, and
2. the code that protects these paths (including resource connection and authentication, authorization, activity logging, data validation and encoding), and
3. all valuable data used in the application, including secrets, keys, credentials, intellectual property, critical mission data, personal data, and
4. the code that protects these data (including encryption algorithms, checksums, hash digests, access auditing, and data integrity and operational security controls)

The practical application requires you to overlay the attack surface model with the different types of users – their roles, privilege levels, etc., that can access the system (whether authorized or not). We can see complexity increases quite rapidly with the number of different types of users, how the attack surface is used for a given purpose, and the set of attack patterns or vectors used. All good reasons as to why we chose to pursue semi-automated intelligent graph searching methods described in later sections.

In our approach, we would report an attack surface as being:

- For each resource in a system (*X* in our notation)
    - a set of metrics characterizing the significance of node(s) within the attack surface *w*, e.g.
        - number of successful attacks (i.e. an attack that invalidated a security property) chains in which the node is successfully attacked
        - histogram of qualitative measures in the attack steps themselves (e.g. CAPEC "Typical Severity" hi/medium/low)
        - per attack, specificity of the attack vs. criticality of the modeled node

By creating derived metrics and visualizations of these base metrics, this approach can inform:

- the system modeler
    - by showing impacts of "fixing" a system, whether by changing the specificity of a model (i.e. from "Router" to "Cisco 1841 Router") or by component selection (i.e. from "Belkin 1234 Router" to "Cisco 1842 Router")
- the attack analyst
    - by enumerating accessible paths through the system
    - by highlighting potentially unknown or unimagined attack patterns
    - by identifying critical nodes

Attack surface depends on the attack patterns, which in terms depend on:

- prerequisites for the attack to take place,
- impact of the attack at each node,
- initial conditions of the system,
- flow of information.

Assessing the Attack Surface: Composable Attack Patterns

An attack surface is a ranked list of the components of a system. The understanding and assessment of the attack surface is informed by the system's requirements and architecture, as selected from the component library. The quality of the assessment is influenced by the robustness and commonality of the attack and component libraries. The rank order of the surface is determined by metric selection at runtime, depending on the user and their needs.

The shortcoming of defining the attack surface in this way is that attack surfaces cannot be trivially compared: i.e. an attack surface can't be aggregated into an "overall security score". For example, two sets of components, connected identically but with different requirements will yield numerically incommensurate rankings. Similarly, evaluating two component configurations with equivalent requirements, but different specificity, will likewise be incommensurate. However, given sufficient

expert insight, two attack surfaces can be compared by inspection, serving as a starting point for architectural design of defenses or expansion of component and attack libraries.

Component Library

To become an effective acquisition technique, the up-front cost of creating a useful model of a novel system-of-systems must go down over time. The most productive approach to driving this is in the creation, curation and tool-based discovery of a library of components-as-architected that employ consistent modeling primitives. To meet the MissionAware goal of assessing the attack surface of a configured system, this means the creation and maintenance of a set of SysML models that consistently employ SystemAware semantics, namely SysML ports and flows carrying Information, and InformationRequirements that employ the Misinformation properties.

Once available as SysML models, the component library can be continuously extracted into the component graph, or "shelf" from which mission systems can be constructed.

System Configuration

SysML provides a means for describing the system-as-configured in the form of instances. In the traditional tooling, this is a user-taxing process, as completing the story of component composition and linkage requires navigating a deep set of UI controls.

In the MissionAware approach, the configuration of instances should be as automatic as possible, enabling the remixing of components into novel configurations while maintaining design intent. Once configured, either through the SysML tool or as part of the War Room process, the system of interest is loaded into the configuration graph, which is the baseline as-configured system.

Attack Library

The Common Attack Pattern Enumeration and Classification (CAPEC) library is a verified third party attack library made by MITRE. The library classifies all attacks under predefined categorizations and abilities of inheritance (e.g., when the attack is possible). CAPEC offers two ways of accessing its libraries: (i) manually through their webpage or (ii) through a downloadable .xml file. Both contain all aforementioned qualities and prerequisites for an attack. The former option for accessing the library is useful to an expert wanting to get specific information for a specific attack. The latter provides the option of extracting those properties in an automated manner through some text manipulation and standard libraries. There are two parts that come in the form of an .xml file: the content itself and the schema. The content contains all the information about all attacks, while the schema defines the categorizations. By "mining" the schema it is possible to get a set of predefined categories one is looking for to apply to the model itself (e.g., for each attack have certain attributes that we know we are looking for and we know we can extract and then map to the actual content). By using the attack library the characterization as well as the prerequisites can be extracted through an automated manner. Hence, it is not necessary anymore to have an expert devise attack trees and assess the threat to the system by inspection.

This could be an optimal solution assuming that all attacks are classified under the same categorization scheme. However, because of the nature of the attacks they have to be classified in different manner. While this is still a finite number, it would not be beneficial to give the model that level of specificity. The problem becomes apparent when one inspects the relationship between the prerequisites and the

library of components, which in terms could, potentially, have a relationship with the impact. To remedy that a certain number of characterizations have to be chosen that encompass most, if not all, attacks. The drawback of this approach is that some information for the attack has to be thrown away during the mining process of the data from the .xml file. Nonetheless, this could still give a better picture of the possible attacks than any expert could by manually inspecting the whole system. This means that a check has to be made after the results from the modeling process are generated. At that point the vast majority of the process of assessing a system for cyber-vulnerabilities has been automated and, then, the experts can work in a more efficient manner.

Assessment Quality

Over time, the quality, reproducibility, and applicability of the attack surface assessment should improve. While increasing the size of the component (L_c) and attack (L_a) libraries may yield more varied results, the largest driver of assessment quality is the relative commonality of the component library and the attack library. This is because the attack patterns must use the terminology provided by the component library to achieve the adversary's goals: with no commonality, few useful attack prerequisites would be met, and this process would yield even fewer impacts.

In the attack library, broadly recognized classifications (such as the CAPEC model) provide the greatest opportunity to reveal emergent adversary behaviors, whatever the size of the library. By contrast, a larger library (such as the CVE) provides the greatest potential to find real, specific attacks. By growing the available classifications more slowly than the size of the library, the overall robustness of analysis is improved.

In the component library, much like the attack library, the richer the hierarchy of component and flow definitions, the more likely it will be to find relevant, interesting attacks against systems defined by these components. A bill of materials is a poor model, while a robust model would include a generalization/specialization hierarchy, like a catalog with headers relating their functionality and connectivity.

Given a large, tightly classified attack library and a rich, hierarchical component library, a degenerate attack surface could still occur if there is no commonality between the classifications in the two libraries. The most productive approach to achieving this commonality would be a guided enrichment, in terms of the classifications from the attack library, of either the component library or the system configuration. Annotation of the component library would be considered authoritative in the future. Annotation of a specific system configuration could be useful, but may not be applicable to all future assessments, and as such must be either a configurable or separate analysis.

Attack Surface:

- Separation of thing that being attacked and the thing that is doing the attack (the impact the attack has on the system)
- Attack surfaces that are not known

Modeling the Mission: The War Room

The MissionAware War Room is a guided, facilitated team activity that leverages the strengths of different stakeholders to make use of system models. Such stakeholders include the mission owner, the system modeler, attack analysts and the acquisition cost experts. By maturing from the "fuzzy front

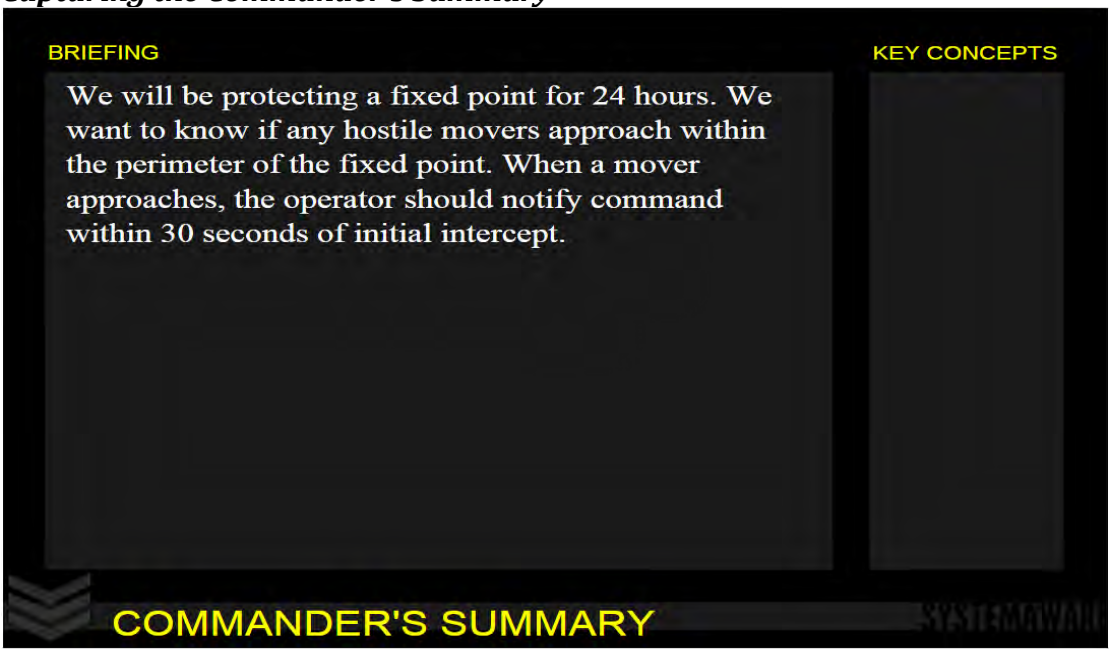
end" of a narrative description to a rough block diagram to a robust system model, model assessment (such as attack surface exploration) can be automated while maintaining linkages to design intent.

While SysML underpins the eventual system model, instead of jumping immediately to innumerable SysML diagrams, the War Room seeks to build the right level of model for the mission at hand in a common language to be arrived at organically by the team.

Team Experience for Decision Support

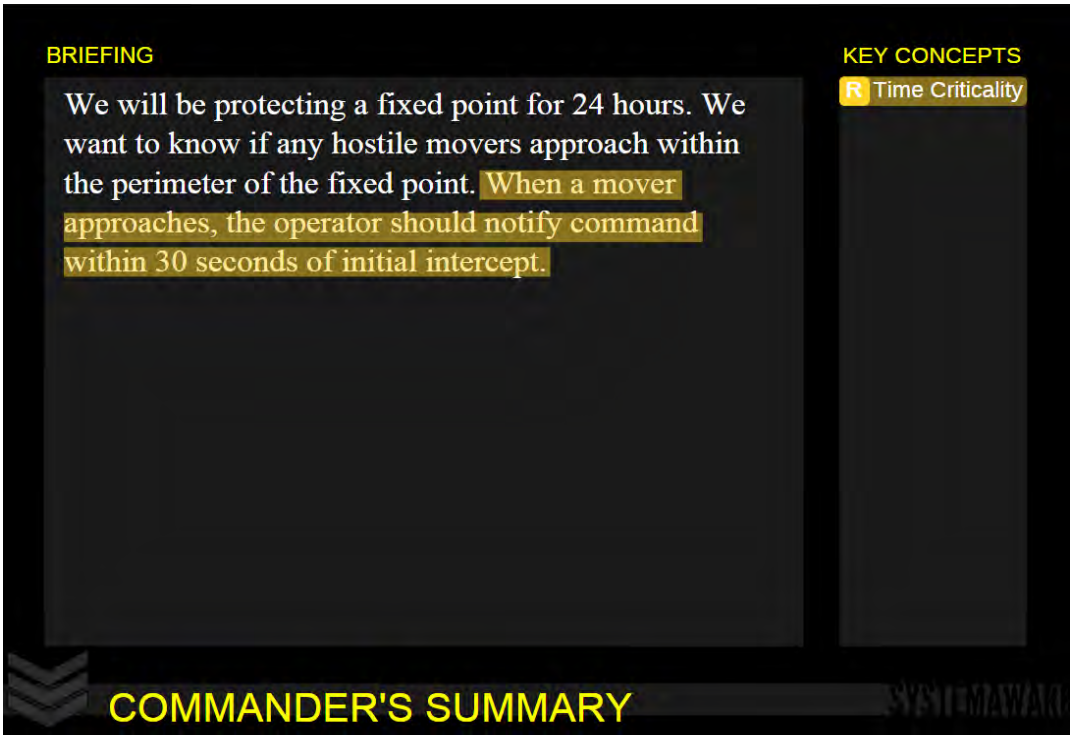
The War Room is structured around a number of screens, suitable for projection in a meeting environment. While presented in a specific order, the steps are intended to be iteratively revisited, enriching the model at various levels.

Capturing the Commander's Summary



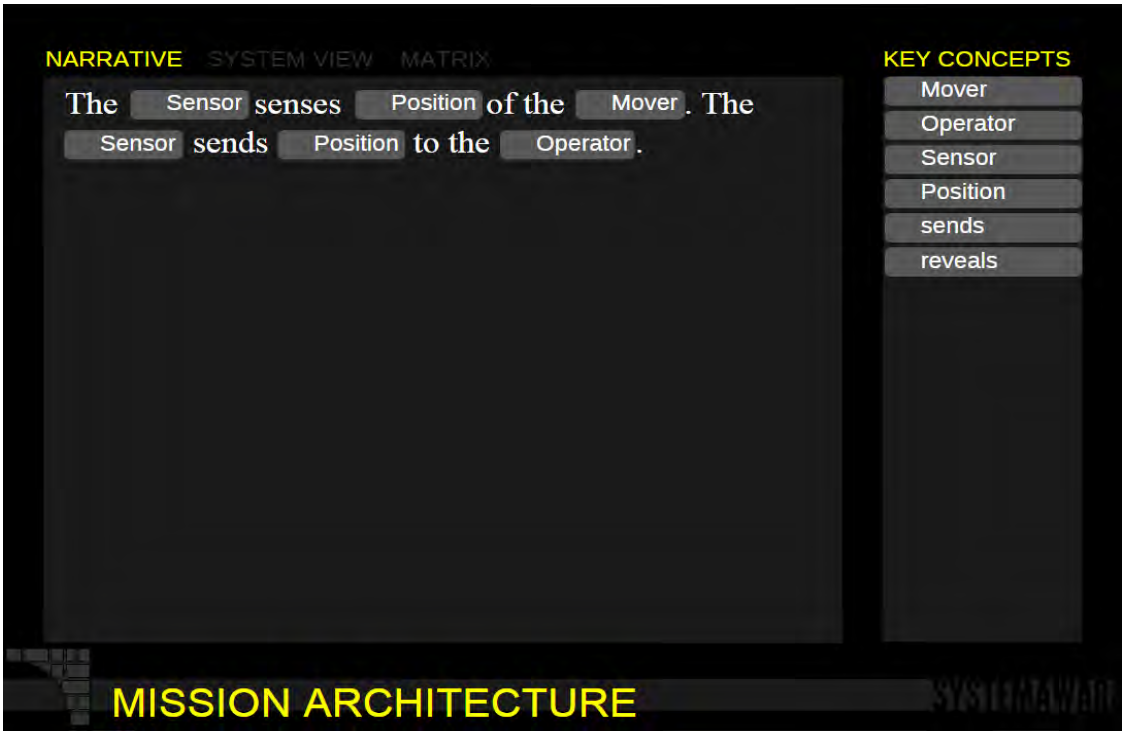
The commander's summary is the high-level, mission-oriented description of the mission.

Maturing the Commander's Summary



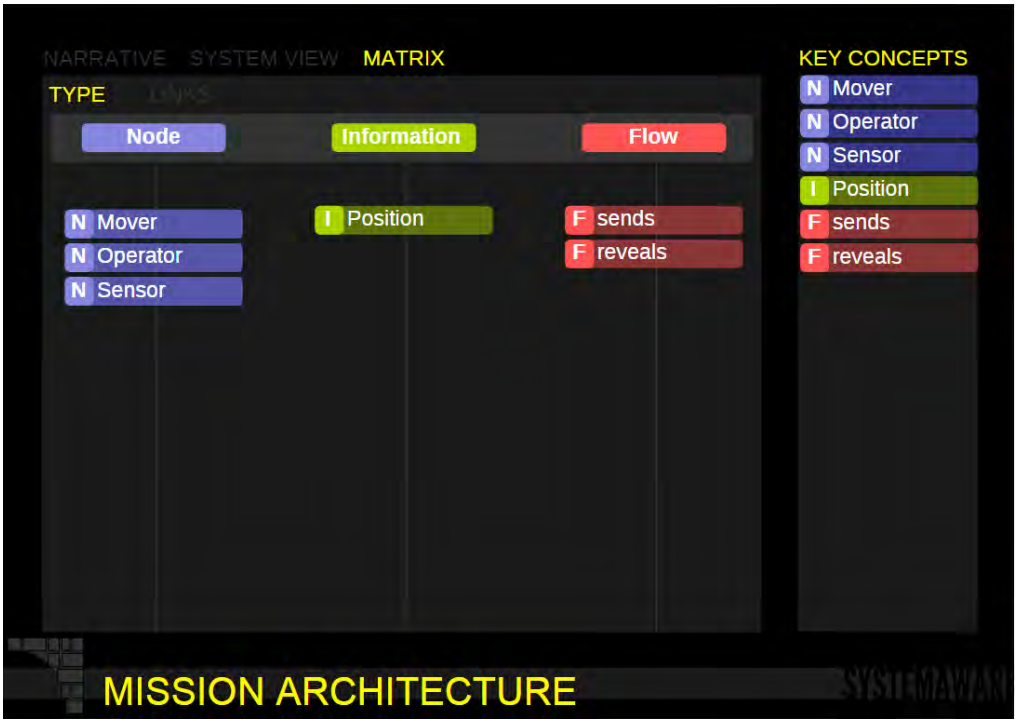
As the commander's summary emerges, the desired behavior of the system forms the basis of mission-level requirements.

Decomposing the System for Information



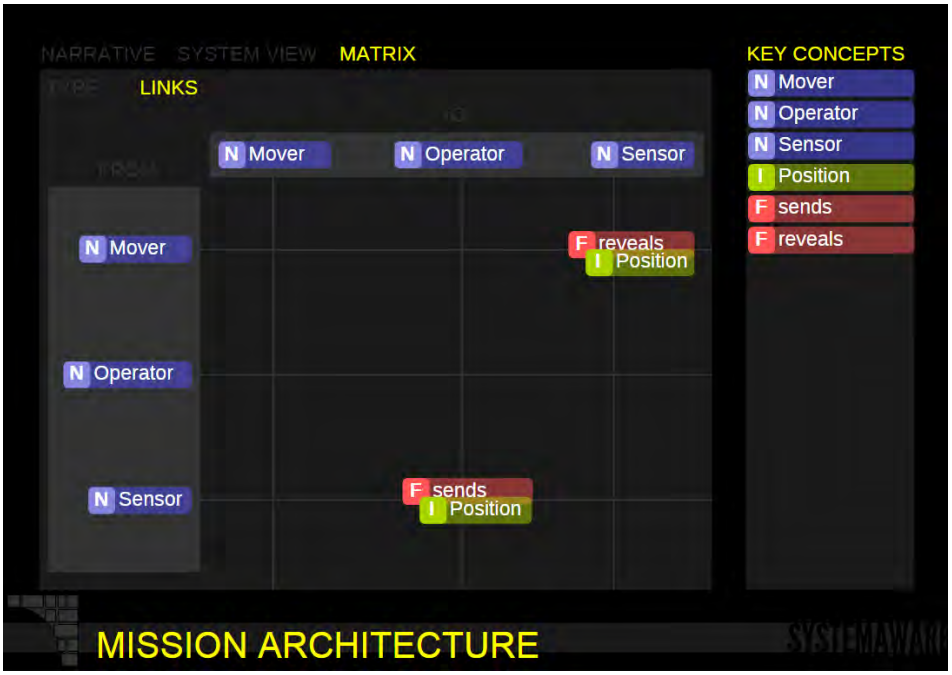
Reapplying the annotation principles from the Commander's Summary, a textual description of information flow through the system creates the framework for the nascent system model.

Categorizing the System



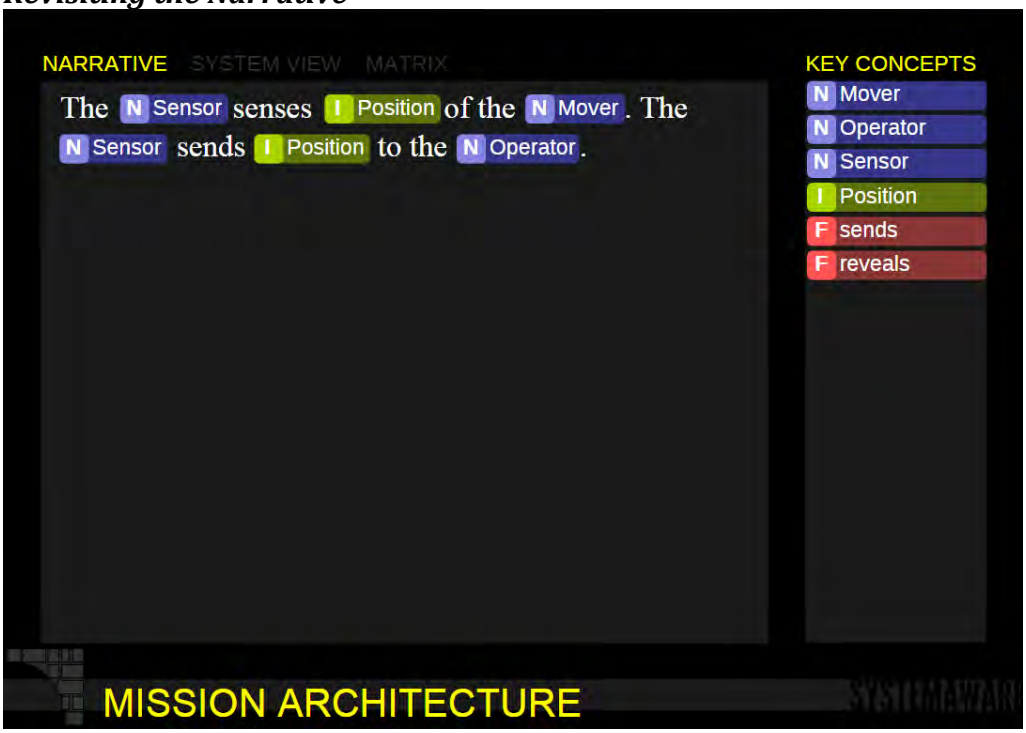
Utilizing a simple three-name system of *Nodes*, *Flows* and *Information*, the system can be decomposed into a form that begins to approximate the SysML level of modeling intent, while not yet focusing on specific component selection.

Connecting the System



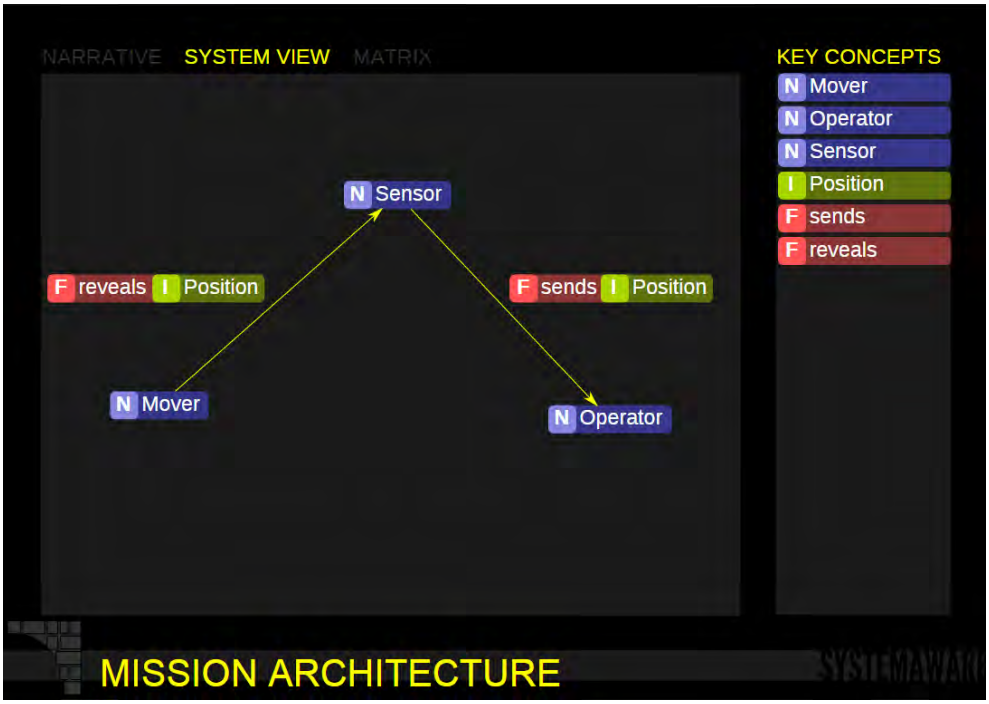
Once organized, the linkage activity of the system becomes much more tractable, allowing for describing the information flow through the system.

Revisiting the Narrative



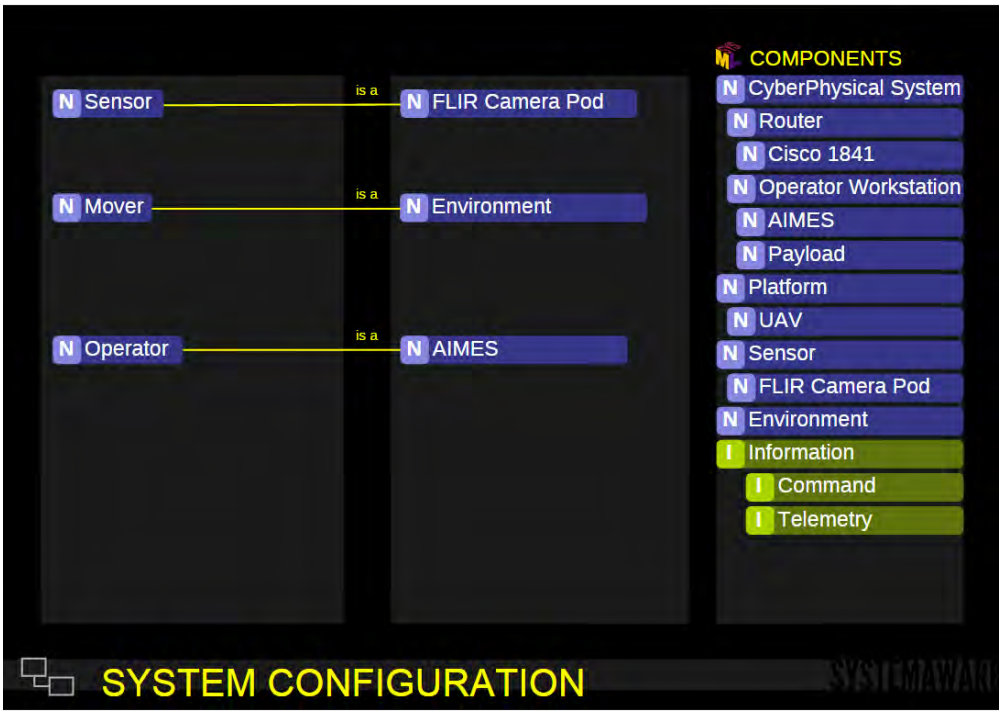
Revisiting, and potentially iterating on the text narrative of the information flow, the stakeholders can observe the interaction of the system.

Visualizing the Model



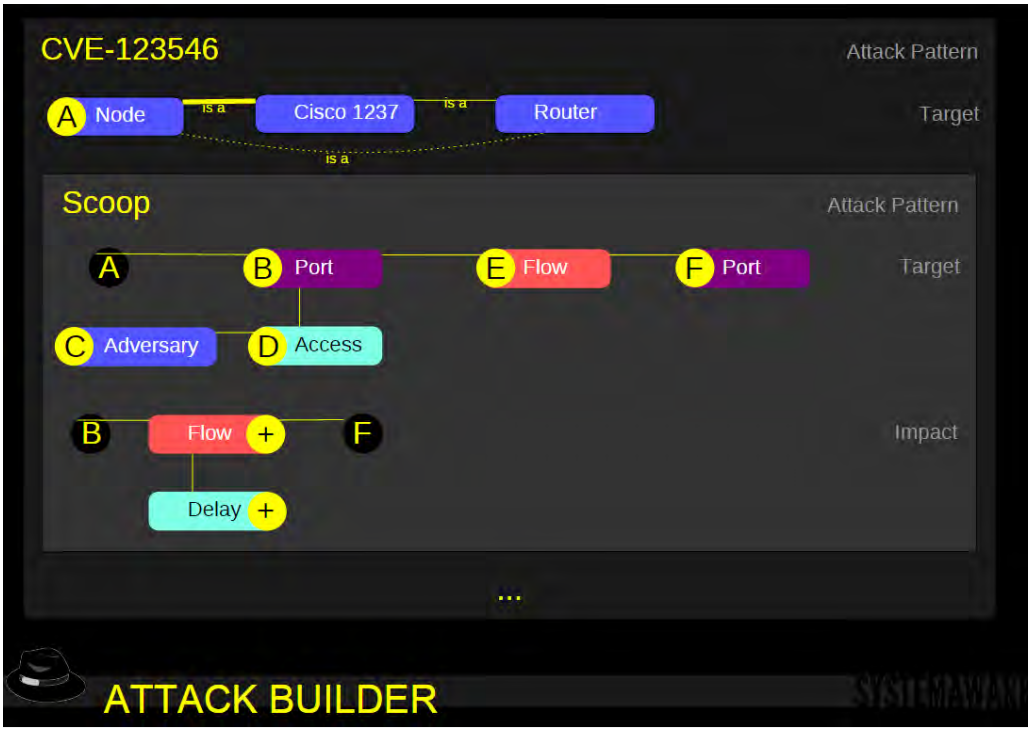
Similarly, the model is now ready to be observed as a more traditional box-and-line description of the system that highlights the functioning of the system.

Configuring the System



Once defined, the model can be further matured to use the language of the component library in a way that provides sufficient information to be attacked via attack patterns. Here, one can choose a highly specific asset, or decide that the component is very general and not interesting to model explicitly. These choices inform the underlying connectivity model.

Attacking the Model



With actual component selections in place, the attacker can describe a set of symbolic relationships that provide fine-grain access to the system model. Attacks can be specified at differing levels of specificity. Combined with the attack library, one is now ready to re-assess the model in terms of its exposed attack surface.

Simulating the Attack: Evolutionary Assessment Tool (EAT)

Our previous approaches to capturing and evaluating the effectiveness of attacks were mostly based on the intuition of the attack analyst, and were custom-built for the particular system of interest. Further, the relationship between a cyber attack and the impact on the functionality of the system was captured in the knowledge of the system modeler.

By enhancing our understanding of both the system-in-context (i.e. mission) and the attack (i.e. attack patterns), we have the data necessary to provide a semi-automated exploration of the attack space. Our initial approach, the MissionAware Evolutionary Assessment Tool (EAT) explores the attack space much like an attack analyst would, considering:

- What part of the system should I attack?
- How should attack that part of the system?
- If that attack is successful, how does it change the system?
- What effect does that attack have on the operation of the system?
- What attack should I try next?

The System and Attack as Graphs

EAT perceives the world of the mission and attacking it as a series of graphs, which contain semantically-rich definitions of various parts of the knowledge required. A given run of EAT requires a component graph, attack graph, system graph and requirement graph.

- Component Graph: *"a **Router** is a kind of **Cyber-Physical System**"*
- Attack Graph: *"a **Router** with an **unsecured wifi port** may be **rooted**"*
- Mission Graph: *"my **router** is a **Router**"*
- Requirement Graph: *"the **link** between my **router** and my **computer** must **not** be **unconfidential**"*

During the course of evaluation, a context graph is created- which contains the state of the system configuration at the current state of the attack.

Genetic Algorithm

Genetic algorithms are optimization algorithms that operate on principles inspired by evolutionary processes to map the MissionAware attack surface.

Organism

In MissionAware, we are evolving an ordered list of attacks of a fixed size. Each attack in a chain remembers:

- Target: the node in the system that is to be attacked
- Pattern: the attack pattern to test, and if all prerequisites are met, apply to the context graph. The "null" attack (of do nothing) is also valid

Each attack receives as an input the current state of the mission as the context graph, and results in either that same graph (unchanged) or a modified version of the graph. A random attack chain is generated by randomly selecting a node from the mission graph and an attack from the attack graph.

Fitness Function

The key aspect of the genetic algorithm is its fitness function, evaluated per organism. As an area of ongoing research, several candidate metrics are:

- **requirement invalidation**: an attack chain that invalidates *n* InformationRequirements is better than one which invalidates *n-1*
- **attack pattern delta**: an attack chain where *n* patterns change the graph is better than one where *n-1* patterns change the graph
- **attack velocity**: an attack chain with *n* "null" attacks before its first attack is worse than one which uses *n-1* "null" attacks
- **attack expense**: an attack chain which incurs a cost of *$n* to the attacker is worse than one which costs *$n-1*
- **attack specificity**: an attack which matches its prerequisites by *n* levels of generalization/specialization is better than one which matches at *n+1* levels

Noting that these are not mutually exclusive, weights are needed. For example, requirement invalidation should usually dominate all other metrics, as it directly coincides with the definition of success an informed adversary would choose. By keeping these metrics in a user configurable space along with graph definitions, the user can "game" the system to reveal unexpected insights. A first pass could be to catch simple, specific attacks, such as when a known CVE will almost always affect a COTS component, by weighting attack velocity and specificity highly. A longer view could be to look not only at successful, requirement-invalidating attacks, but value overall graph change, anticipating "unknown unknowns".

Population

Once a given organism can be evaluated and weights have been selected, then two organisms can be compared; or more broadly, a population of organisms can be ranked. The population is a set of some pre-determined size of organisms. Initially, these are unrelated to each other, consisting of randomly chosen values.

Tournament

The very first population is evaluated for fitness and ranked via the weighted values. From this, the top-performing half of the population is preserved and the bottom-performing half discarded.

Mating

This missing half of the population is filled in with a mating of the remaining population. Simply, this means taking two attack chains and generating a third attack chain that is a mix of its two parents.

Mutation

Finally, to inject a little more entropy, every new member of a population has a small chance to have a particular attack replaced with a new, random target-pattern pair. This ensures that a little "new blood" is injected into the population.

Convergence

In evolutionary algorithms, it is desirable to have a "right answer" possible and quantifiable. In the case of MissionAware, one could argue that invalidating a single requirement is sufficient for an attack to be considered a success. However, as a mission evolves to be resilient to the possibility of information attack, invalidating a single requirement might no longer signal a true failure, and further evaluation should continue to see if more requirements can be invalidated, or more possible paths to invalidation are found.

Assessment

At the end of an evaluation, we can observe the best-performing organisms over the course of the evaluation. Additionally, the final population can be seen as a snapshot of the attack surface, where by simple counting each node in the system graph can said to be in more or fewer successful attack chains than another node, providing a relative ranking of the criticality of nodes.

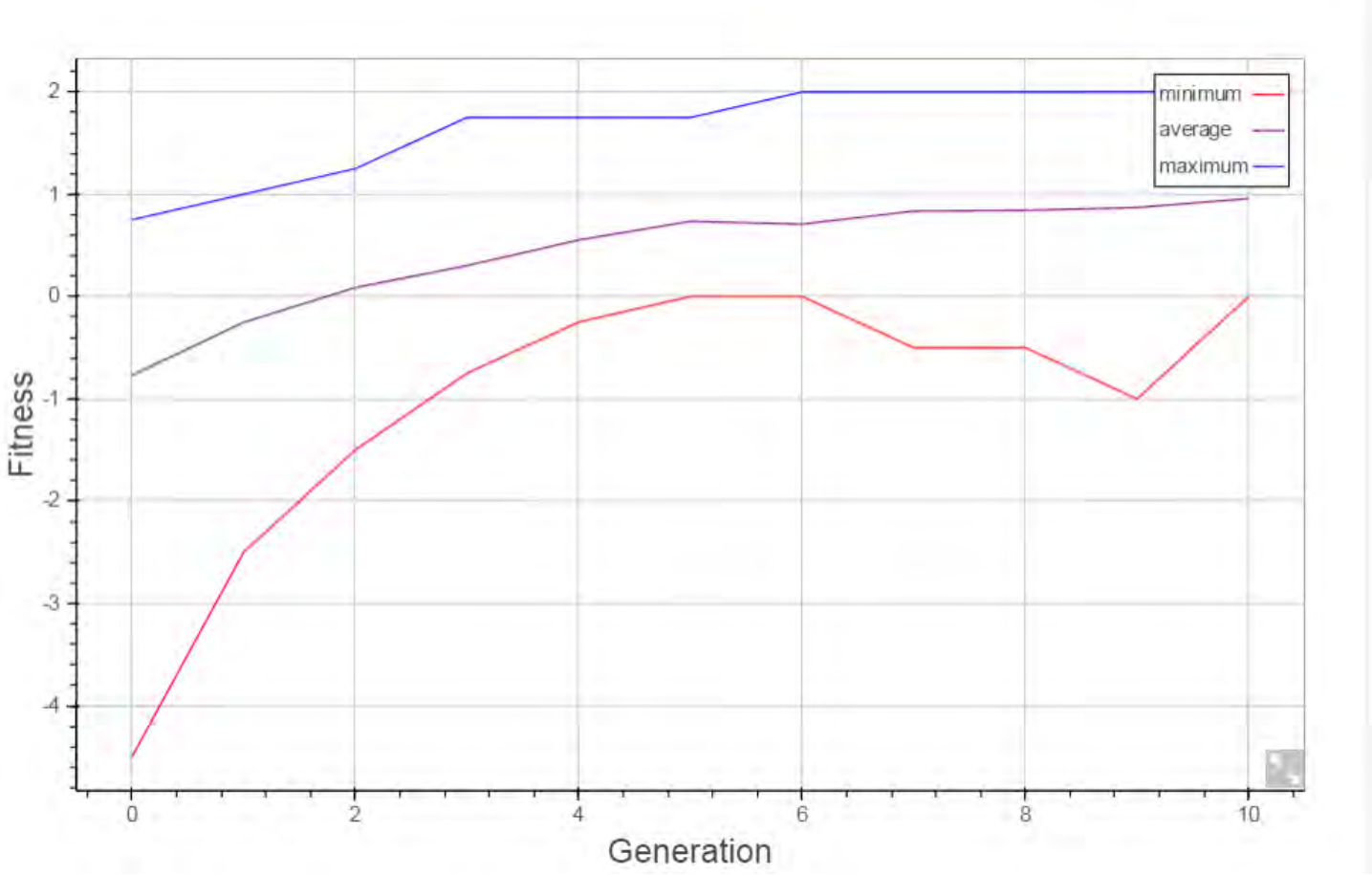


Figure 24 Fitness as a Function of Time

This simple assessment (in Figure 24 above), weighted to reward successful attacks and penalize redundant attacks, shows the fitness of the population improving over time.

Case Study: UAV Autopilot ISR Mission

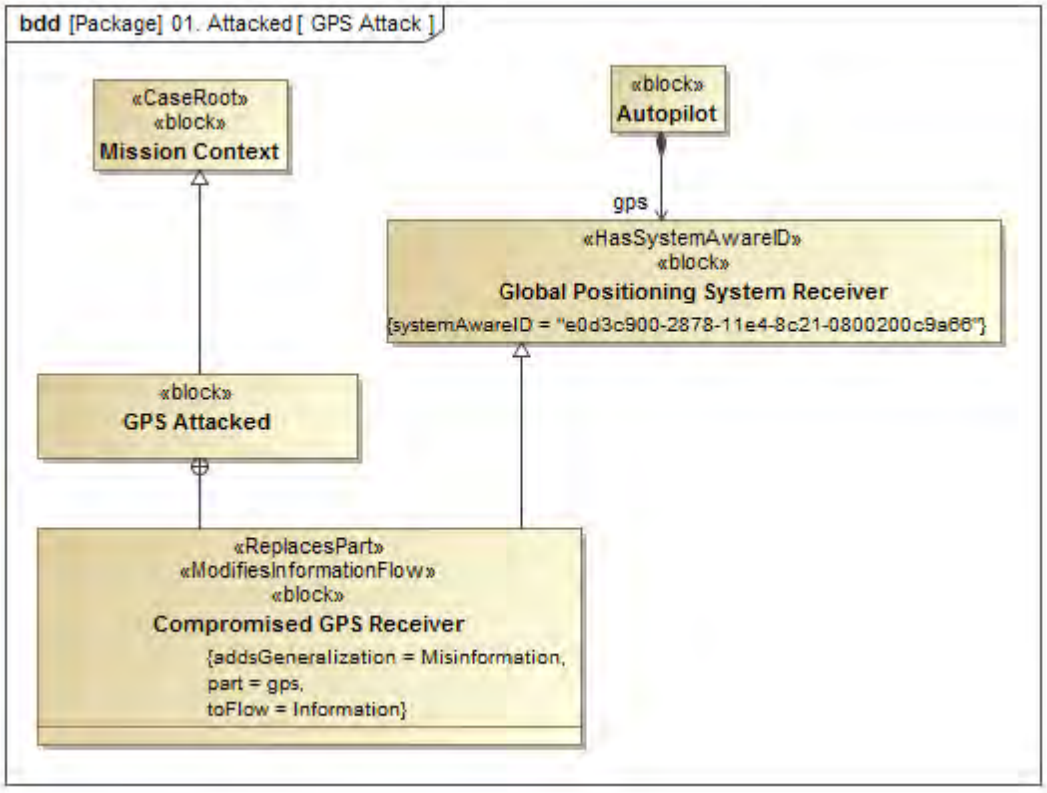
Mission Overview

In SystemAware 2.0, the work revolved around the definition of a notional Intelligence, Surveillance and Reconnaissance (ISR) mission involving a number of operators and a semi-autonomous Unmanned Aerial Vehicle cooperating to fulfill an intelligence requirement. A SysML model of the baseline system was created, with varying levels of detail, with specific interest on the notional COTS autopilot.

Within this context, a supply chain attack was proposed, introducing a compromised Global Positioning Receiver (GPS) receiver in the UAV autopilot subcomponent. In SystemAware 2.0, we modeled the presence and identity of the attack, without providing a concrete way to describe the actual information flows that would be impacted.

Modeling the iterative process of system definition and discovery, we introduced a deeper insight into the system-as-designed, namely a second GPS receiver

Attack Surface Modeling

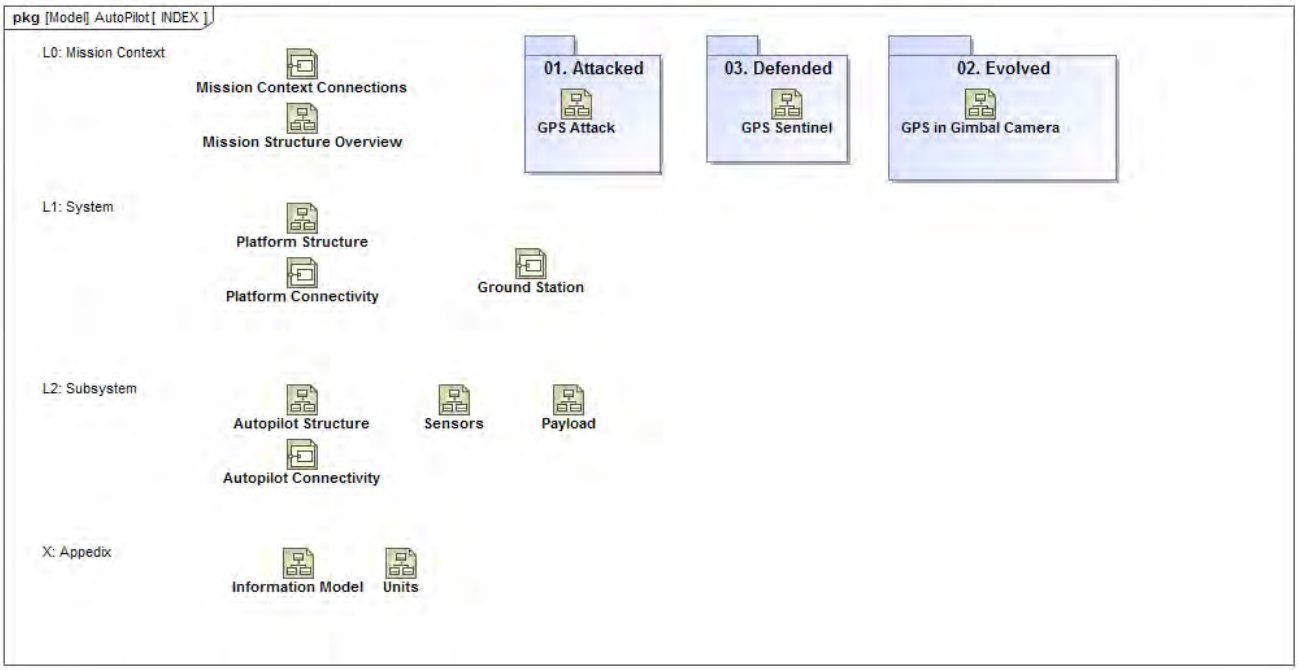


Here is a SystemAware 2.0 representation of a supply chain attack targeting the GPS of a system.

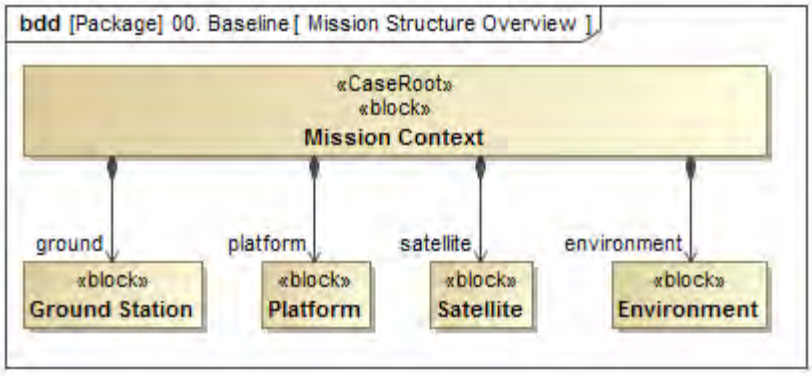
```
id: adv:CompromisedGPS
a: s4sa:AttackPattern
prerequisite:
- exists:
    as: target
    ofType: shelf:GPSReceiver
    ports:
        as: port
- exists:
    as: flow
    from:
        as: port
    toA:
        type: s4sa:Environment
        as: environment
impact:
- add:
    as: flow
    hasProperty: s4sa:ReducedIntegrity
```

Reformulated into the parlance of MissionAware attack patterns, this GPS attack would be reformulated to be more specific about the type of effect the mission would have, as well as more general to find all possible types of GPS receivers.

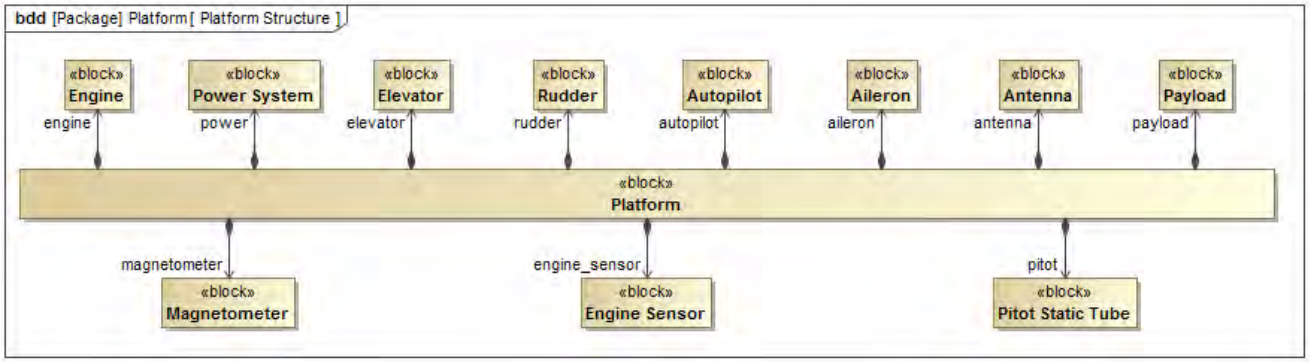
Mission Modeling



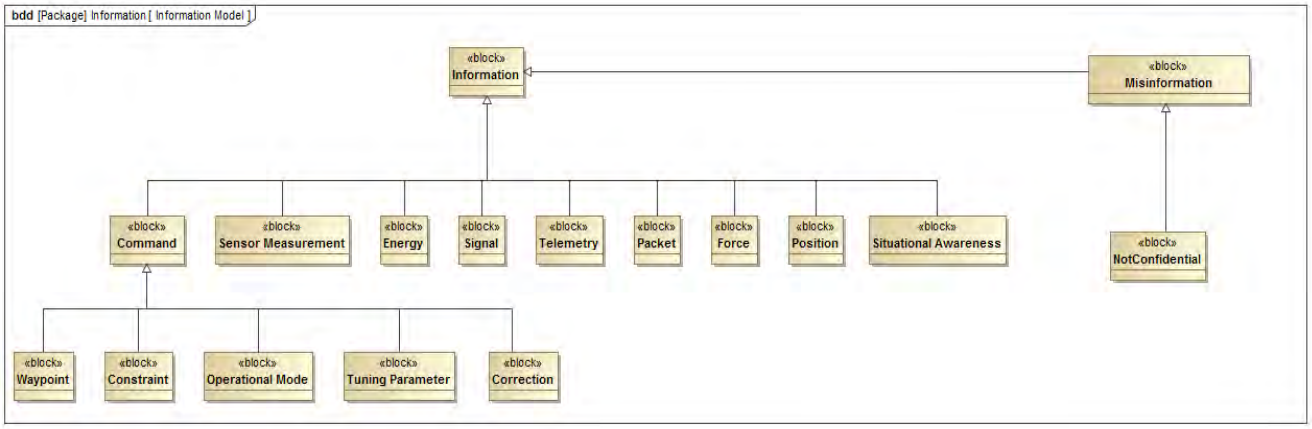
A SysML model is organized by packages, and revealed through a controlled number of types of diagrams.



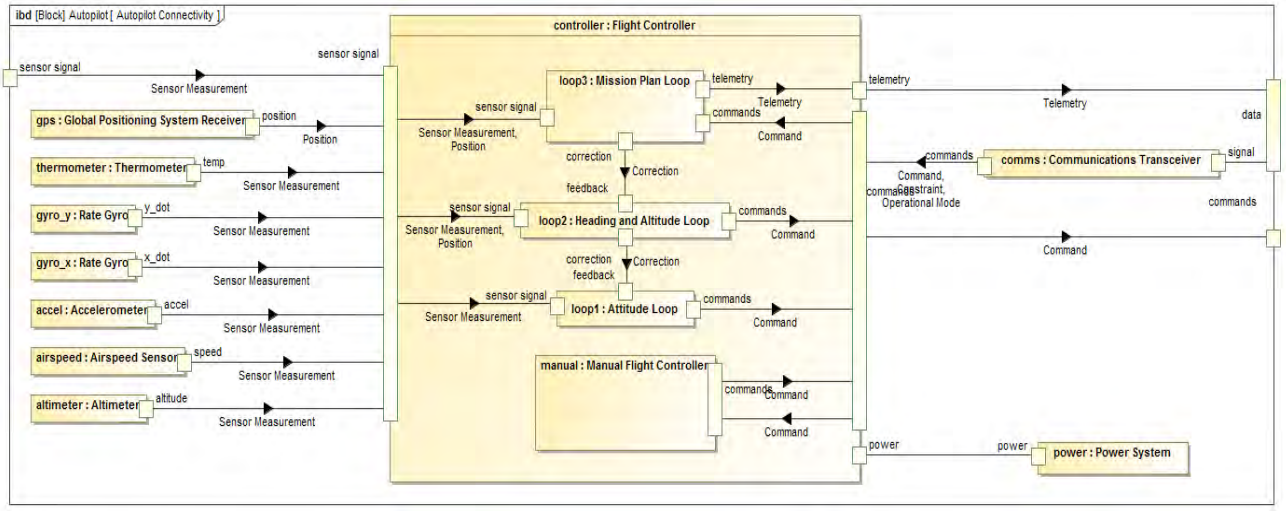
A high-level *block definition diagram* of structure provides context for deeper levels of insight.



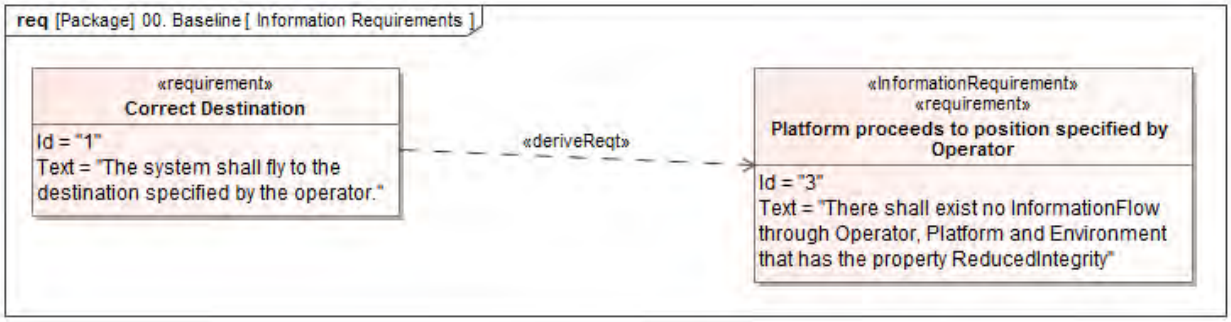
At each level, consistent diagrammatic notation is used, with this example of the platform component.



Block definition diagrams can be more specialized, such as this specialization tree of Information.

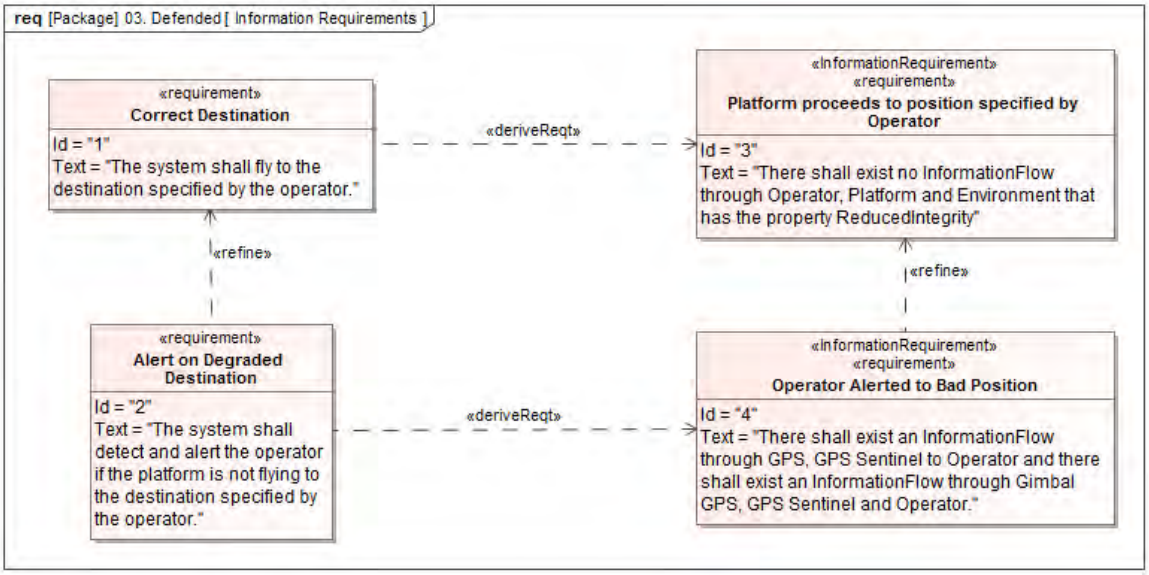


An *internal block diagram* shows even deeper relationships between parts, specifically the internal and external linkage of a system. The values on the edges show the type of Information transmitted.



A requirement diagram shows both a mission-level requirement, as well as a more formal InformationRequirement.

Although this section displays and demonstrates SysML diagrams involving last year's UAV project as a representative example, Appendix 8 shows SysML diagrams for this year's *Base Defense System.*



As a system evolves, its requirements may change as well. Here, an alternate operation mode is captured as another set of requirements.

Findings and Lessons Learned for MissionAware

Many of the model-based systems engineering tools focus on the concrete, viewable properties of a system; namely, energy, mass, cost, etc. Due to the architectural (versus operational) focus of these tools, mission modeling is necessarily low-resolution. Within these tools, reuse in an executable sense is achieved by having component-level and attribute-level compatibility, i.e. machinability of an individual part, or the fuel economy of an engine based on overall system mass, etc. However, when the focus of the mission changes, these models become brittle or otherwise don't scale to reusability across an entire organization.

By broadening the focus of the user experience to include more stakeholders beyond the system modeler and reasoning with existing knowledge of a component and attack library, the MissionAware War Room increases the productivity of model-based acquisition. These increases, realized over time, are enabled through capture of a mission's required flow of information with a consistent modeling approach and continuous assessment of the attack surface of a mission configuration. These features are in turn enabled by the SystemAware SysML profile, providing a baseline for description of Cyber-Physical systems, the Composable Attack Pattern for capturing the otherwise unmodeled or un-modelable weakness of a mission, and the Evolutionary Assessment Tool, providing immediate insights into the evolution of secure mission architectures.

Table 4 (below) shows this portion's compliance with Part I, Section 4 of the SERC document:

Explore whether synergistic use of advanced tools can provide greater assurance levels	Yes. The attack library and evolutionary assessment tool developed in this effort are designed to provide greater assurance to decision makers that defenses protect against the full space of possible attacks

Determine if such tools can be embedded into the decision processes in a manner that facilitates 'what if' analysis	Yes. The practicality of application in basic architectural selection and in the 'what if' analysis was the principal motivation behind the development of the new set of tools. The new methodology provides much more automation and support to the decision maker than did the SysAware tools.
Consider conducting a workshop to expose potential analysts to these opportunities	A workshop with 10th Fleet was conducted late in the last phase of the project. Based on analysis of the results of the workshop conducted early in this phase, it was determined that new tools were needed to support stakeholders and decision makers, particularly in the early phases of the architectural selection process. As a consequence, the corresponding portion of this phase's effort focused on the development of the MissionAware War Room environment. By broadening the focus of the user experience to include more stakeholders beyond the system modeler and reasoning with existing knowledge of a component and attack library, the MissionAware War Room increases the productivity of model-based acquisition.

Table 4 Compliance with Part I, Section 4 of the SERC Research Topic document.

References For this Section (Architectural Selection and Assessment)

[1] F. Yarkochkin, "Hunting in the Shadows: In depth Analysis of Escalated APT Attacks", BlackHat Conference 2013, Las Vegas, Nevada

[2] NIST 2013 Workshop on the Foundations of Cyber Physical Systems, prepared by Energetics Corporation. http://www.nist.gov/cps/

[3] A.A. Cardenas, T. Roosta, and S. Sastry. "Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems". Ad Hoc Networks, 2009.

[4] GAO. Critical infrastructure protection. Multiple efforts to secure control systems are under way, but challenges remain. Technical Report GAO-07-1036, Report to Congressional Requesters, September 2007.

Summary

The various sections of this report detailed the four (4) main components of the CY2015 research effort:1) the development of multi-sentinel architectures, 2) the consideration of various Human Factors issues, 3) a demonstration of how Cloud-Based Sentinels (along with the resultant integrity monitoring) could be performed, and 4) an assessment of how Advanced System Modeling and Attack Tree Tools can be integrated into a cyber security assessment workflow.

Although these sections have been written by different authors, we feel that these topics dovetail together nicely into an overall discussion of enhancing cybersecurity for complex *Systems of Systems* that are so often encountered in our increasingly complex world—and the authors sincerely hope that the techniques and technologies discussed here help to bring about improvements into this realm. Appendix 12 contains a separate report on defending a Software Reconfigurable RADAR (SRR) topic that was added to this RT-136 later in the fiscal year.

Appendix 1: List of Abbreviations and Acronyms

ADF	Automatic Direction Finder
AFRL	Air Force Research Lab
AIMES	Advanced Intelligence Multimedia Exploitation Suite
CONOPS	Concept of Operations
DIA	Defense Intelligence Agency
DoD	Department of Defense
DoS	Denial of Service
FMV	Full Motion Video
fNIRS	functional Near Infrared Spectroscopy
FTP	File Transfer Protocol
GPS	Global Positioning System
GTRI	Georgia Technology Research Institute
GUI	Graphical User Interface
HTML/CSS	Hypertext Markup Language/
HW	Hardware
IDS	Intrusion Detection System
IP	Internet Protocol
IR	Infrared
IV	Independent Variables
LRMS	Long Range Motion Sensors
MQ/RabbitMQ	a message queuing system used by Rabbit Technologies for the RabbitMQ server
NGIC	National Ground Intelligence Center
OMERF	Observatory Mountain Engineering Research Facility
PICTE	A manager for FMV video used by SAIC/Leidos
PIR	Passive Infrared
RC	Radio Controlled
REACT	Resources for Early & Agile Capability Testing
RFID	Radio Frequency Identification
RPAS	Remotely Piloted Air System
SAM	Sentinel Alert Message
SBC	Single Board Computer
SIEM	Security Information and Event Manager

SIGINT	Signal Intelligence
SSHFS	Secure Shell File System
TTL	Time To Live
UAV	Unmanned Air Vehicle
UDP	User Datagram Protocol
UGV	Unmanned Ground Vehicle
UPS	Uninterruptable Power Supply
UVa/UVA	University of Va
VM	Virtual Machine
VSOM	Video Surveillance Operations Manager (from Cisco)
XML	Extensible Markup Language

Appendix 2: Camera and Sensor Overview

The purpose of this appendix is to summarize the project findings, especially the findings obtained by midsummer 2015. Our main objective was to design and implement stations of cameras and sensors (and a means to display intruder events) in order to simulate a cybersecurity system. The team had to overcome various learning curves and had faced a few difficulties along the way. The results, however, were successful in terms of meeting the goals of project and of a 'working demonstration' by the end of the project. In that, the team was very successful.

Major Project Components for the "Sensor" component

These technology areas were explored and exploited to provide workable solutions to the research effort:

A) Video streaming and modifying frames
B) Researching different types of sensors and implementing them
C) Setting up the RabbitMQ/UDP server and integrating with the sensors/Base Defense system
D) Developing the Base Defense web application
E) Constructing a USB plug-in/UDP cyberattack and defense

Video streaming and modifying frames of images

Objective: To stream video feed from a PiCamera (a camera module for the Raspberry Pi) over the network as well as to modify the pixels of the frames in the middle of the data flow

In addition to the four (4) Cisco camera video feeds monitored separately, we were able to stream video from a Pi Camera ("PiCam"; a camera attached to a Raspberry Pi SBC) as well as modify the incoming pixels within the video frames, an ability used by the team to trigger certain attacks. From there we started with a video stream, extracted individual frames, modified them, and then formatted them back into a video. We made use of a library called OpenCV to get the individual frames and to modify them. By working extensively with video technologies, we were able to learn a

great deal about video standards, media frameworks, pipelining protocols, and even a little bit of networking.

Researching different types of sensors and implementing them

A wide variety of motion and acoustic sensors were investigated by scouring the web to locate the most appropriate sensors in terms of cost and features. The team settled on a diverse suite of sensors, some of which were interfaced with the Raspberry Pi and some of which were interfaced through Arduino SBCs (with the goal of achieving hardware diversity). Additionally, some sensor data was transported over 'hard wired' Ethernet connections while some data was transported wirelessly (demonstrating path diversity).

Setting up the RabbitMQ/UDP server and integrating with the sensors/Base Defense system

Our objective was to utilize the RabbitMQ and UDP servers to transmit sensor data from the Raspberry Pi and Arduino to the Base Defense web application (again, for path diversity), The team developed a method of sending out messages from both SBC types (Pis and Arduinos) via these different paths, and this was very successful. See Appendix 4 for a schematic of signal paths, with hardware names and IP addresses.

Developing the Base Defense web application

Since the previous sentinel project used the *Django* web framework for their sentinel web app and its increasing popularity in general use, the team decided that *Django* would naturally be a good framework to use for the *Base Defense* module. By using Django, the team could easily transfer or mimic some of the logic from the previous project into *Base Defense*. It also made it much easier to learn Django when the team had a full example to look at while trying to figure out how the framework functioned. In addition, Django is based on *Python* which plays nicely with RabbitMQ and UDP and which we used for the majority of work.

The *Base Defense* web app was to consume messages from the RabbitMQ servers and the UDP stream in order to populate the app with detections. Therefore, the team wrote a *Rabbit consumer* that consumed messages from the Rabbit server and, for every message, requested a URL from the Base Defense app which created a new detection. Likewise, the team wrote a *UDP server* that received UDP messages and queried the web app for the URL which created new detections. When we got a new detection, we'd simply create a new marker, which would drop onto the map (a 'bird's eye image of our reactor room, taken from high above the room). In addition, it would add the marker to a list of all markers below the map. The team then built two columns (instead of one) below the map, which separated out the UDP and Rabbit detections. With these changes we had more structured and straightforward functionality than before. By extensively working with web frameworks and technologies, the team was able to get hands-on practice with Django, JavaScript, and HTML/CSS.

Constructing a USB plug-in/UDP cyberattack and defense

Objective: To simulate a cyberattack against a sensor station (Pod 1 in the demonstration) and a detection scheme to defend against the attack

The team developed two strategies:

(1) To configure a USB *plug-in* to enable a cyberattack of cutting off the communication link between a sensor station and the *Base Defense* web application. The detection scheme involved sharing the directory that held the relevant source code with a remote machine that monitored all the important files in that said directory. Upon detection, it would replace the corrupted file with a backup file stored in the remote machine.

(2) To set up a UDP client that transmits erroneous sensor data to the *Base Defense* web application. To correct for this attack, the logic just disabled the UDP sensor that was associated with the erroneous UDP messages. In a more sophisticated scenario, the Sentinel would have built-in logic and sensors to assess the situation more appropriately among different types of hardware. For example, if there were three sensors (the two for Base Defense and an extra one specifically for the Sentinel) installed with three data streams flowing into the Sentinel, it would be able to execute a more informed decision by opting out the "outlier" in a two-out-of-three voting scheme. This project component proved to be very rewarding, as the team was able to learn about the utility and power of bash scripting and SSH/SSHFS. This 3rd sentinel was indeed installed on two (2) pods.

Appendix 3: Midsummer AIMES Overview

The AIMES server ingests video feed slices that PICTE sends it. Metadata is extracted, the video file is moved to a permanent location on the server, and values in the database are updated to contain key metadata values, and the location of the slice file on the server. The server also responds to web requests from *Exploit*. The most important is to get a list of the active feeds and the multicast addresses where these feeds exist. In this way, the server acts as a remote. It points the exploit client at the feed, and then the client starts listening to this address. The Exploit client is used by an operator to view and interact with active video feeds (being broadcast from PICTE) or with archived video files (being provided by the AIMES server). When the operator requests information, like the active feeds or archived video slices, requests are sent to the AIMES server for processing.

The AIMES Server is responsible for receiving slices, frames, and important mission data from PICTE and storing this data into the Postgresql Database. The Server runs in a Windows Server '08 virtual machine. One of the crucial functions of the AIMES Server is to supply the ability to look at archived video files. These video files need to transferred from PICTE to the Server, and in many systems these two components are operating on the same machine. In this case the video transfer simply happens through a file transfer within the same machine.

The process of taking slices from PICTE involves a complex combination of python scripts that strip the metadata from the stream, then take the metadata and inject specified fields into an XML template, standardizing the data. This standardization allows for another program to easily parse the xml for ingestion into the desired database tables. This process is crucial to how active feeds and archived video files are accessed from a client using Exploit, and their importance is emphasized in the section below concerning a Metadata Attack on the AIMES system.

AIMES Attacks Overview

Replay Attack

This attack involves altering the address Exploit looks at to receive its video feed in order to replace the correct feed with a looped replacement (e.g. replacing regular footage with footage of an empty room). For the purposes of this report, the attack details have been removed.

When the attack commences and the operator searches for active feeds, they will see *pod1* and not *test* in the list; however, when they open the *pod1* feed, they will see the replayed video coming from *test.*

Video Quality Degradation Attack

Similar to the replay attack, this attack involves altering the address Exploit looks at to receive its video feed in order to replace the correct feed with a degraded version of the actual one. Video degradation is accomplished by capturing the correct video feed and replacing one byte at the start of each UDP packet with an arbitrary value. This results in the video being rendered unusable but still recognizable as the correct feed (e.g. very fuzzy, colors are altered, movement is difficult to discern, but larger shapes still show up as a general outline, etc.).

Metadata Attack

As mentioned above, metadata taken from a video stream sent to the AIMES Server goes through a series of python scripts in order to ultimately comprise the entirety of the entries into the Postgresql database.

Crucial details have been removed for the purposes of this report, but the Metadata attack works by shutting off the server, removing one of the standard python scripts with our own malicious python script. Then the server is restarted and the malicious script is compiled and run on the server. The modifications made to the python script are simple, there is code that takes the metadata from the video stream and converts it to XML, when this occurs we have access to all of the values and are able to offset the date timestamp by as much as 5 years

Correct implementation of the attack will cause all video archived (after the attack has taken place) to be logged incorrectly in the database with a faulty timestamp. Specifically, 5 years previous to the correct time stamp. This archived video correctly plays and has the right time stamp associated but is very hard to search for (due to the erroneous timestamp). Other metadata values can be corrupted in this way, just alter the right script and include that script in the attack

"The Trigger"

The *trigger* isn't an attack in and of itself; but it is a way to set off attacks already in the system at any given time by altering pixels in the video frames being ingested into PICTE. It involves having code on the same network as AIMES (or whichever system is being attacked) to monitor the video being sent to PICTE, searching for a predetermined pattern in the video (e.g. certain pixels being changed to specific values) and executing one of the attacks described previously, as opposed to manually setting off the attacks.

AIMES Attack Protections

AIMES Alert Protections "Side Car":

The team wrote a standalone GUI to provide additional functionality to the AIMES exploit client operator. To use the *Side Car*, set up the means to monitor a feed by connecting to that feed in *Exploit.* When you begin monitoring a feed, updates are sent to the sentinel about its metadata to ensure that the feed you're watching is not being tampered with. Relevant updates from the Rabbit server are displayed in the GUI, these currently are just alerts about the feeds that you are connected to. In the future, there may also be updates from the sentinel about which feeds to be looking at, or even confirmation messages to make sure that observed actions are actually coming from the operator

AIMES Protections Developed by the Team

Metadata Ingestion Script Hashing

This is intended to protect against Metadata corruption originating on the server by editing the ingestion scripts. The team collected hashes of the 4 ingestion scripts mentioned in the description of our metadata corruption attack and stored these values on the sentinel (the Ubuntu VM on the

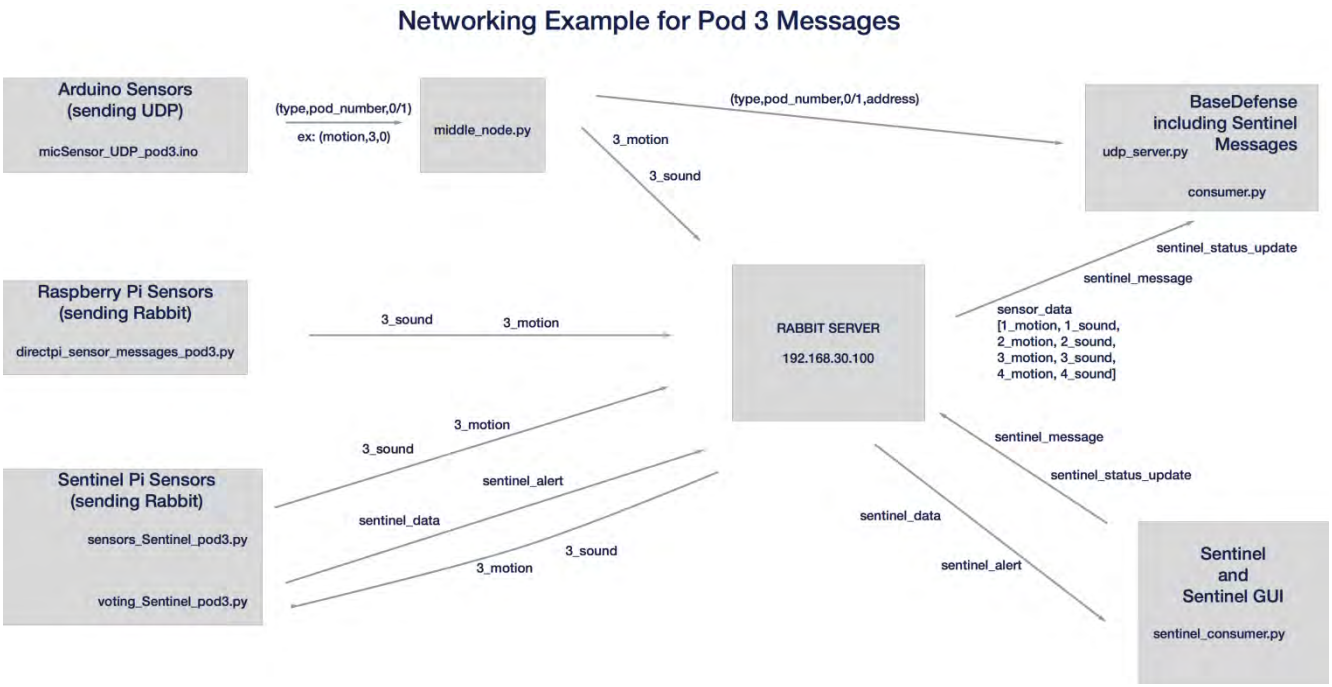
Alienware laptop). The sentinel runs code that listens to messages from this queue and compares the values to the expected hashes. It is important that the rabbit messages be properly formatted, as the sentinel is expecting the hashes in a certain order and flags the operator if the hashes are not what was expected (and in the correct order). This protection only prints to the console if there are errors; it would be possible to alter the code so that it sends the proper Rabbit update messages if so desired.

ISRCrossCue Data Cross-Checking

This is a fairly in depth protection that relies on Exploit's ability to write one metadata value to a file on the AIMES server. This file gets updated in real time, one per feed that is currently being watched. If the stream is legitimate then the coordinates in the file will match the expected value (because the sentinel is keeping track of the pattern).

If the stream has been tampered with, the coordinates will not always match the expected values and will trigger an alert to the SideCar GUI. If a client tries to connect to a feed with incorrect coordinates, an alert is given. And if a client submits incorrect codes for any reason while watching a previously verified legitimate feed, an alert is given.

Appendix 4: Message Flows Between the Various *Base Defense* Components



Appendix 5: A Suggested "ALERTCON" Scheme for Operator Attention Based on Sensors Triggered

As mentioned in the main document, the team worked on a suggested "ALERTCON" scale that would attempt to raise operator awareness (and potential concern) based on the quantity of sensors being triggered; and to some extent, the TYPES of sensors being triggered. This ALERTCON scale would range from 0 (no alerts) to 6 or 7 for many sensors triggered, many of them 'local'. A strawman range is suggested below:

A "Standard" Event sequence starts with one of the Long Range Motion Sensors (LRMS) tripped OR the RADAR detects motion, and then other pod sensors get activated as the intruder approaches

The most notable reaction: LRMS turns on some floodlights, and either one the LRMSs or the RADAR alerts operator(s) to approaching intruder in a "DEFCON" like sequence of ascending numbers (0-6 or so).

First message on *Base Defense* : "**Long Range Sensors Activated**" (see ALERTCON 1 below)

Reaction: Operators assume a higher level of vigilance by watching video feeds and *Base Defense* display for events on the screen (eventually, if the intruder proceeds inbound, the motion is detected by the pod's PIR motion sensor, audio voice pickup by the microphones, etc.) and an operator compares incoming messages on the *Base Defense* display and also the video displays on AIMES and also the Cisco IP cameras

In this scenario, the Sentinel monitors the various message traffic and the incoming message inputs from all sensors and compares this activity with its own PIR motion sensors (mounted on pods 3 & 4 sentinels) to determine the following DEFCON ("ALERTCON") levels as the intruder progresses:

ALERTCON 0: no activity detected, anywhere (every few minutes, *Base Defense* display prints "**ALERTCON 0**" (in green) out to the screen as a positive indication of being "alive" (a 'heartbeat pulse')

ALERTCON 1: activity detected on one (or both) LRMD sensors AND/OR the RADAR message printed out to the screen as a "Sentinel" message): "**ALERTCON 0.X: A Long Range Motion Sensor Activated**" (message is in blue with a grey background, as below:)

ALERTCON 1.5: "motion detected on one of pod__PIR motion sensors w/o an initial LRMS activation"
ALERTCON 2: "motion detected on one of pod__ PIR motion sensors **with** an initial LRMS activation"

ALERTCON 3: "motion detected on both of pod_ PIR motion sensors w/o an initial LRMS activation"
ALERTCON 4: "motion detected on both of pod__ PIR motion sensors **with** an initial LRMS activation"

ALERTCON 5: "motion detected on both of pod__PIR motion sensors w/ an LRMS or RADAR activation"

ALERTCON 6: (and flashing) "motion detected on multiple pods___ PIR motion sensors, and three of the LRMSs with RADAR"

The sequence above (from ALERTCON 0 through 6) represents a more or less 'normal' progression of messages that would occur with an intruder being detected (initially) by the LRMS and/or RADAR devices and then more pods (pods 1-4) start detecting motion(s) as the intruder approaches the building. The location that these 'alertcon' messages show up in is up for discussion; perhaps along the bottom of the standard Base Defense display, perhaps on the sentinel display (the 2nd of 3 displays). The color of the message and the size of the text help denote urgency or relative importance of the message.

If ONLY audio is detected by a pod, a message should pop up on the screen (as a Sentinel message) that denotes this fact—that **audio** has been detected **without any motion being detected**, and for the operator to check the video camera feed that is directly opposite of the pod that detected the audio noise for activity (pod 3 is opposite pod 1, & pod 2 is opposite from pod 4, etc.). This message should be in dark yellow or orange in an attempt to garner attention.

If the team can establish such a path, perhaps operator tags typed into the AIMES system, as operator 'notes' or 'items of interest', can flow out of the AIMES system (as a message) and into the Sentinel. The Sentinel rebroadcasts this message (merely as a 'pass thru') to the Sentinel message display on *Base Defense*. This serves as a means by which the AIMES operators can flag suspicious items in their video feed & pass that to whoever is monitoring the Sentinel messages on the *Base Defense* display, which would be important IF the AIMES displays are not collocated with the *Base Defense* system and these messages could perform a useful *tipping & queuing* function.

The team can certainly build in additional logic if needed, but this represents a good start--- and one that can be built upon. The overall idea is that these *Base Defense* messages compliment the standard UDP/IP and RABBIT messages in a way lets the operators know that either 'normal' operation seems to be occurring, or that there may be something that is amiss and resulting in conflicting results from the various system monitors.

Appendix 6: Sample Messages if Sentinels Start Detecting Conflicting Data

The following text outlines the proposed "CYBERCON" levels, the colors used for the 'message box' (green, yellow, or red) that denote the condition that the Sentinel thinks the entire system is in, from 'green' (normal), to 'yellow' ("I think something is going on"), to 'red" ("Yes, something is definitely not adding up"). These individual messages are not broken out or implemented yet and need to be reviewed for utility. They are not TRUE 'sentinel' messages that denote any sort of attack on the system, they are merely flagging odd behavior that may indicate a cyberattack.

CYBERCON Level:	Light Color	Printed Message	Implied Condition
0	Green	"No Cyber Attach Detected"	all is well
1	Yellow	"Audio Heard With No Nearby Motion Detected"	(self-explanatory)
2	Yellow	"Motion Sensed Near Pod X with no LRMS or RADAR activation	" " "
3	Yellow	"Motion Sensed on Only One Half of pod 2 without other PIR being activated. Possible Cyber Attack"	(pods 2, 3, and 4 only)
4	Yellow	"RADAR activated with no other sensor activated. Check displays for activity"	
5	**Red**	AIMES Operator flags malicious activity viewed near pod (X), but no motion is detected in this area. Possible Cyber Attack."	
6	Red	(for pods 3 & 4 only) "Motion Sensed on only one half of Pod (X) without other sensor being activated. Sentinel PIR sensor is ACTIVE. Definite Cyber Attack on Pod (X) PIR sensor due to 2 out of 3 voting that enhances cyberattack detection	

Appendix 7: The Sensor Pod Sentinel System and Protection Against USB Device Insertion

The purpose of this document is to detail the sentinel protection against a possible USB thumb drive insertion. Our main objectives were to develop another web application for the Sentinel system and to scale as well as to integrate the USB plug-in cyberattack and defense, titled as follows:

A. Developing the Sentinel web application

B. Scaling the USB plug-in cyberattack and defense

Developing the Sentinel web application

Objective: To develop a web application for the proof-of-concept, *system-aware* Sentinel.

Strategies & Challenges: The functionality of the Sentinel web application was very similar to the *Base Defense* web application. That is, the purpose was to provide another layer of alerting capability to the users of potential cyber threats. Furthermore, the Sentinel hardware itself (i.e., additional

Raspberry Pi's introduced to the sensor pod system) was designed to be modular so that it could adapt to systems other than our system of pod sensors.

Therefore, the team had to integrate another Raspberry Pi to pods 3 and 4, which were designated to be the "sentinel" pods. This design decision was made in order to compare and contrast between the sentinel pods and pods 1 and 2 that did not have a Sentinel system integrated. These additional Raspberry Pi's were almost identical to the existing Raspberry Pi's in the pods, i.e., they gathered motion and sound sensor data and passed them off to our central server. The additional functionality of these Raspberry Pi's (or sentinels), however, was that they were fed with motion and sound sensor data from the other two existing sources (Raspberry Pi and Arduino) in order to perform voting with its own motion and sound sensor data. These sentinels then alerted the Sentinel web application if it detected an inconsistency in the data stream.

Since the Sentinel web application was very similar to *Base Defense*, the team was able to hit the ground running very quickly. At this point, the team had a pretty good understanding of how to develop applications such as *Base Defense*. The only major feature that differed was the integration of the USB plug-in cyberattack and defense. In other words, the Sentinel app needed the capability to detect sentinel alerts coming from pod 3 or 4 as well to execute the corresponding countermeasure. Detecting the alerts was something that the team was already familiar with, which involved making simple http-requests. The countermeasures, on the other hand, involved running bash scripts designed to run a particular set of Unix commands to carry out the actions of replacing the compromised source file with a golden copy. This particular feature of the application was the most challenging because the team had to learn how to synchronize background processes (via bash scripts) with the Sentinel web application, which then also had to invoke more bash scripts when appropriate. Integrating the USB plug-in cyberattack and defense into a web application was certainly a learning highlight. The end result of the system was a success, as this portion of the project played a significant role in the demonstration.

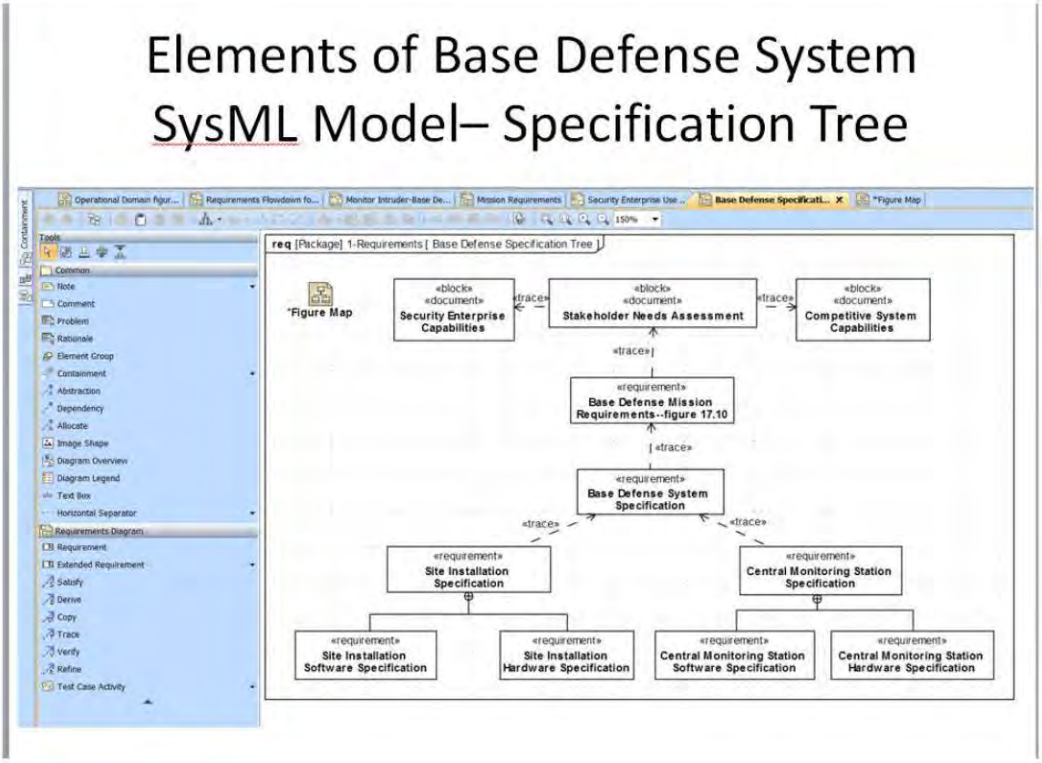
Scaling the USB plug-in cyberattack and defense

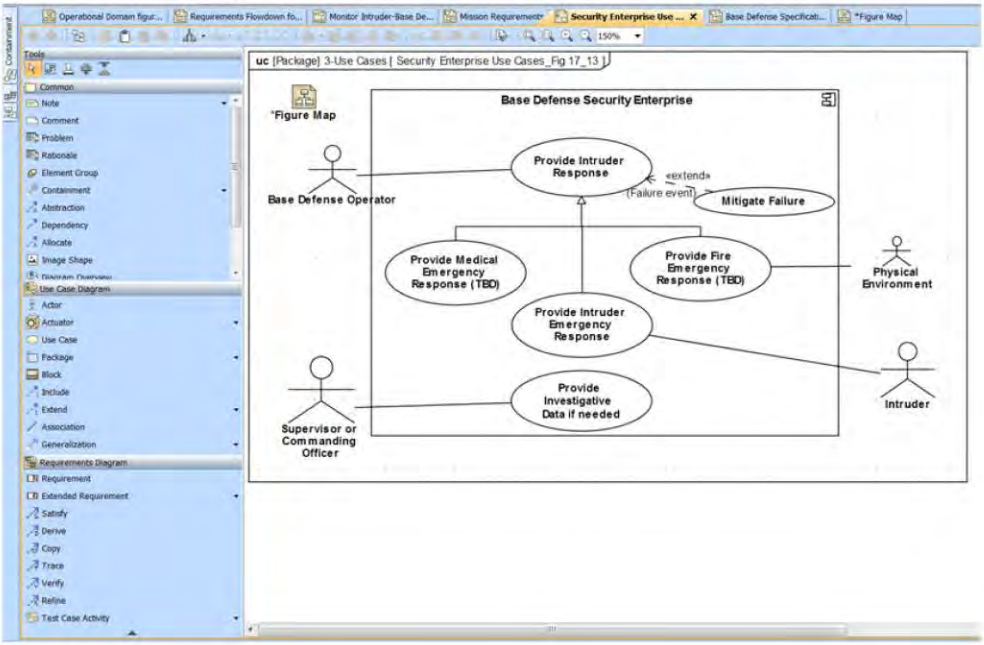
Objective: To scale the USB plug-in cyberattack and defense from one pod to a system of four pods. For more information on the implementation of the USB plug-in cyberattack and defense, please refer to the previously discussed plug-in attack

Strategies & Challenges: At a high level, the plan was to have a single source that monitored the files of the four pods in parallel. To that end, the team designed a file system that organized the remote connections, and this was the first challenge because we were not positive if was possible to secure more than one remote connection. After confirming that it was, the next challenge was to ensure the synchronization of the system because we were running many processes at once. This part of the development required careful design because the team soon ran into many dependencies between the processes running via bash scripts. Unfortunately due to time constraints, before the team was able to fully flesh out the system, we resorted to implementing it for just pods 2 (without sentinel) and 3 (with sentinel) for the purposes of the demonstration.

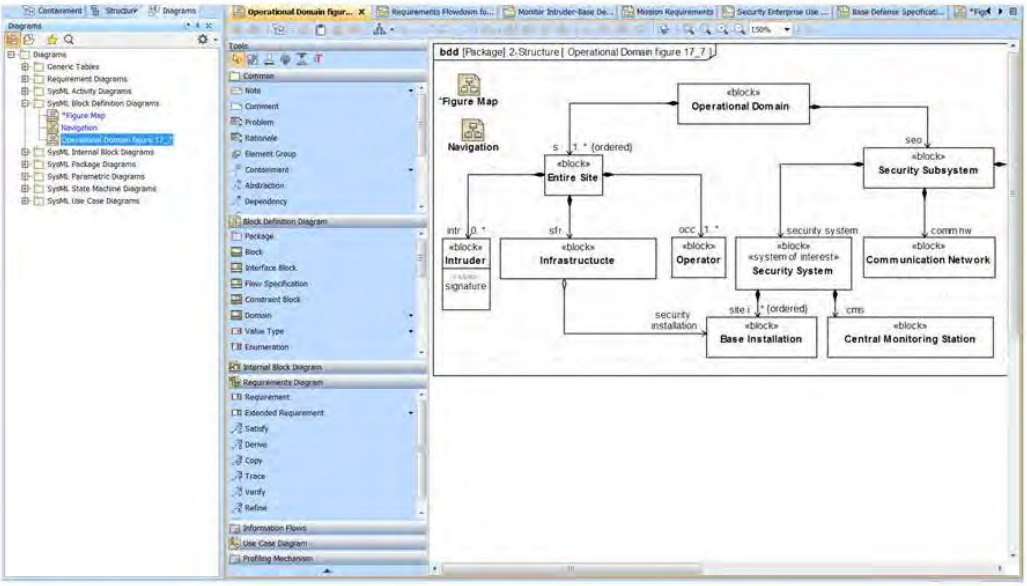
Results: Although the project was not fully scaled, it was scaled up to a practical point, which ultimately illustrated the concept fairly well at the demonstration on September 18th, 2015.

Appendix 8: Sample SysML Diagrams for the Base Defense System Prototype

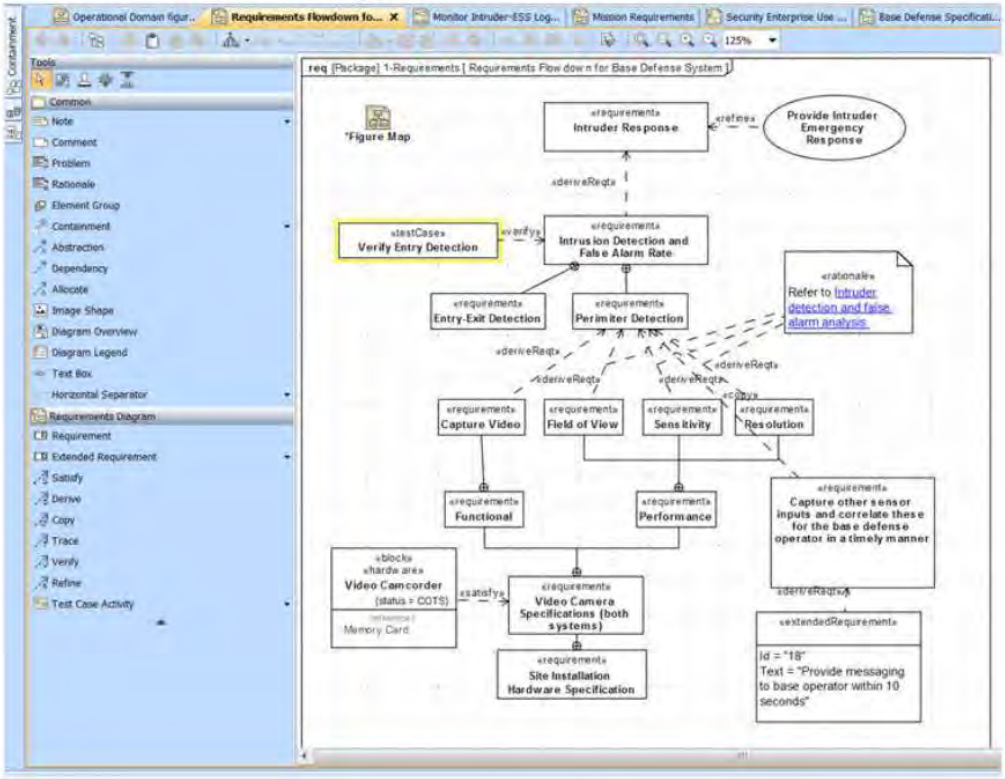




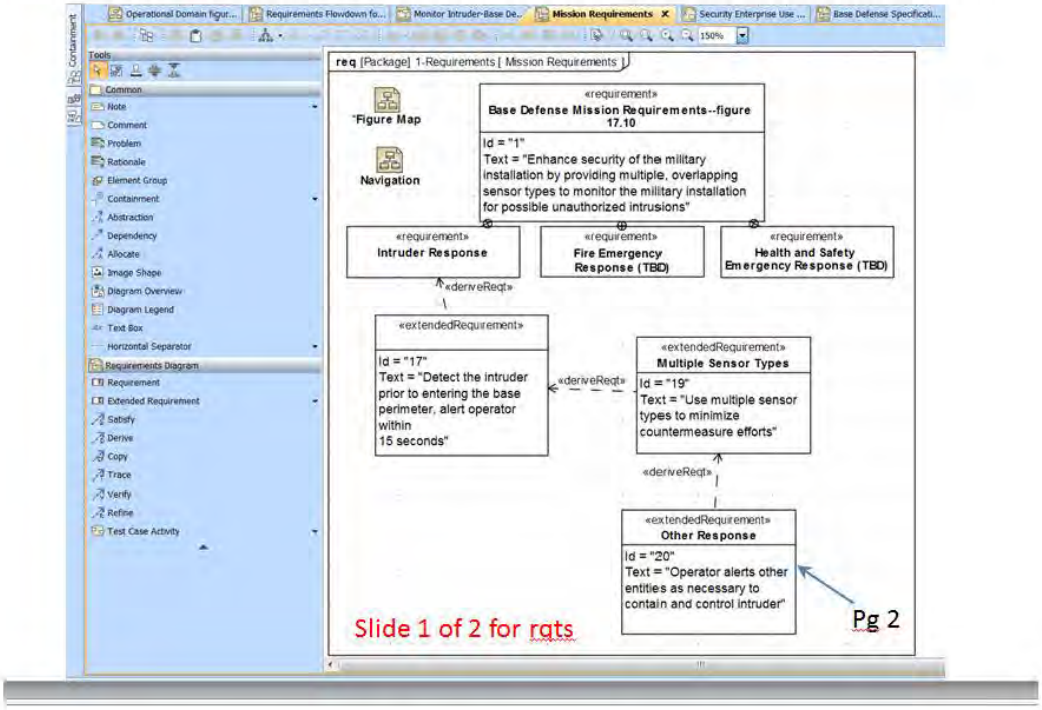
Elements of Base Defense System SysML Model– Operational Domain



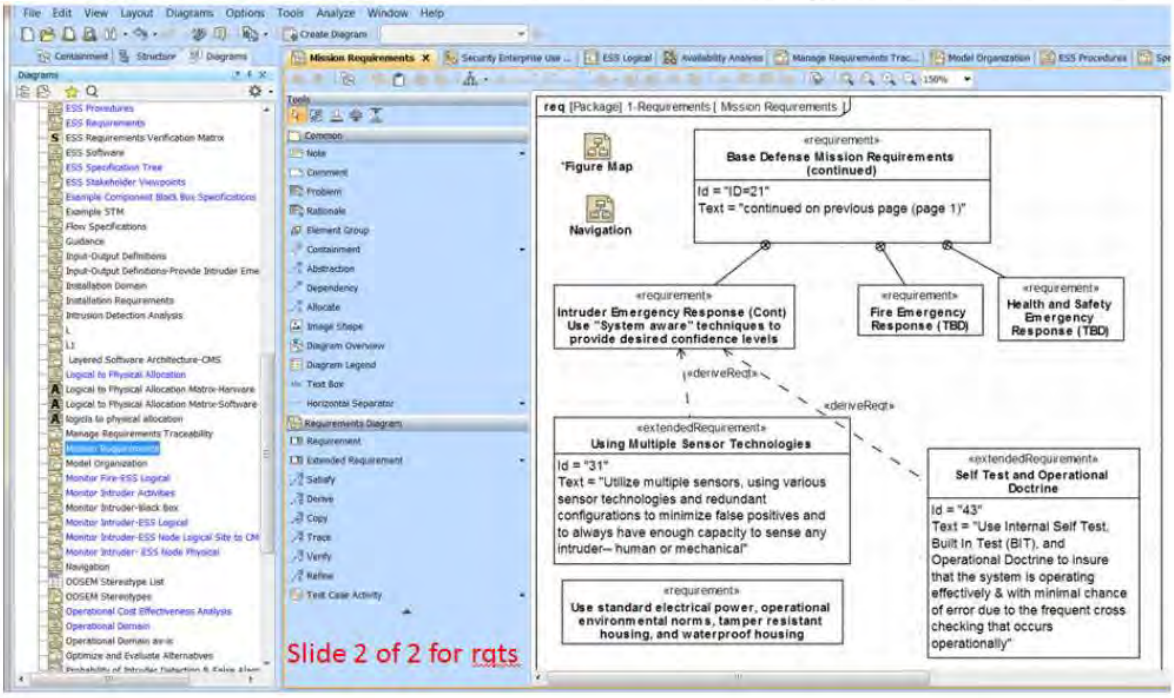
Elements of Base Defense System SysML Model– Requirements Flow Down



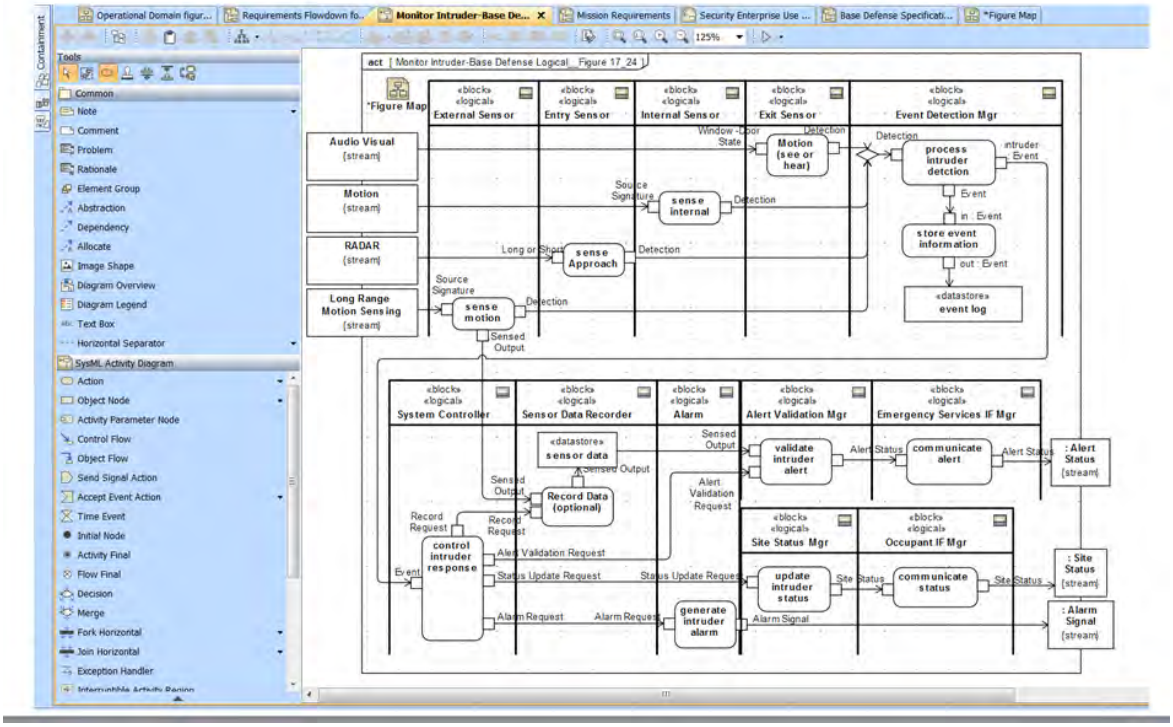
Elements of Base Defense System SysML Model– Mission Requirements



Elements of Base Defense System SysML Model– Mission Requirements



Elements of Base Defense System SysML Model– Monitor Intruder Logical



Appendix 9: Formal Definition of Attack Surface and Attack Patterns

An attack surface is roughly the set of system's actions resources which can be modified via actions (attacks). The more extensive the attack surface is, the more vulnerable the system can be. Prerequisite attributes for an attack surface to exist are:

- Susceptibility - Is the system attackable, critical functions that lead to significant damage.
- Accessibility - Can the system be reached, what are the channels or means for Ingress

An attack surface of system Q is the tuple $w = (X, L_c, I, M, O, Ch)$

Where:

L_c *is defined as set of all components*.

In practical terms, L_c is a predefined library of components for a given mission as instantiated in a SysML model of system Q.

Let *w* be*:*

- X is a subset of components (resources) taken from L_c that comprises system Q. X_j is an instance of a component from L_c that is a resource in system Q.
- I is the complete set of inflows (entry points) for all of X. I_j is specific inflow for a component X_j.
- O is the complete set of outflows (exist points) for all of X. O_i is a specific outflow for a component X_j.
- M is function that maps X, I, *and* O to *images* of X, I and O.

$$M: X \underset{i}{\rightarrow} \widetilde{X}, \quad M: I \underset{i}{\rightarrow} \widetilde{I}, \quad M: O \underset{i}{\rightarrow} \tilde{O}$$

M defines the transformation of X, I and O in terms of integrity, confidentiality, timeliness, and availability properties. These are considered the "secure" states for Q.

- Ch is a function that assigns to each entry or exit point of a component X_j an attack pattern A for all of X *and* $\tilde{X}$ in Q

$$Ch: I_j \cup O_i \rightarrow X$$

$$Chs: \tilde{I}_j \cup \tilde{O}_j \rightarrow \tilde{X}$$

Appendix 10: Potential Ways to Attack the System

The following list shows a sample of hypothetical attacks that could be mounted against the system, including an estimated level of complexity in table 5 below.

a) Take measures to avoid detection as you approach the protected installation/building under protection

b) Inject spoofed data into the Ethernet stream (or WiFi stream) to raise alarms, cancel previous messages, or to swamp and confuse the system

c) Inject spoofed/fake messages to/from the Rabbit server, even with client authentication and encrypted exchanges (not sure on the encrypted exchanges)

d) Perform a Denial of Service (DoS) on the Wifi (or even the Ethernet stream) to prevent ANY data from getting through, but this would be easily detected in most cases

e) Inject discrepancies (real or artificial) between what the A side & B side sensors see on the platforms, to trigger sentinels and to cause confusion

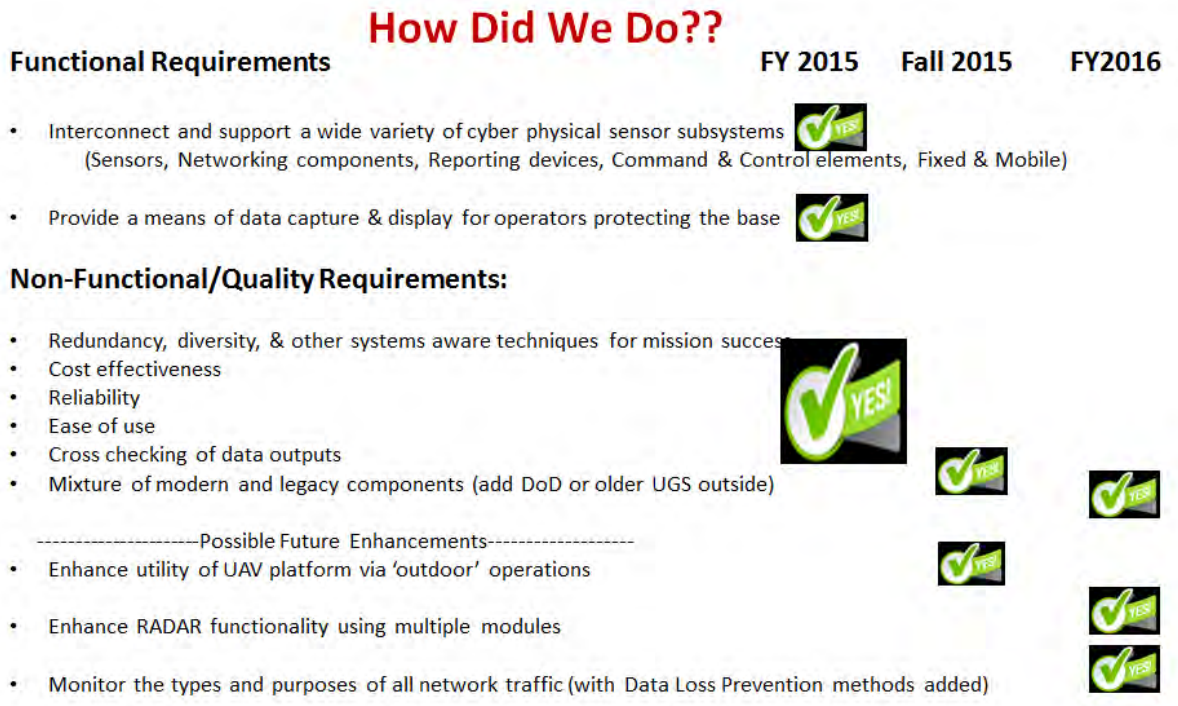
Attack	Type	Level of Complexity (1-10)	and explanation
a	Cloaking	10	due to our multiple sensors and sensor types
b	Integrity	6	detect these false messages by using a hashes on legitimate messages
c	Integrity	6	same as b above
d	Availability	4	this results in no message traffic, which should be easily detected
e	Integrity	8	Difficult to do with A side and B side integrity/sentinel monitoring

Table 5 Type of Attack Category and Level of Complexity for Attacks a thru e

Thus far, 'confidentiality' is not a major factor, but could be easily implemented by encrypting the message traffic using data link encryptors to hide the contents of the messages. The team deems this easy to implement, if needed.

Appendix 11: End of Project Report Card for Operational Features

The following chart was presented at the Sept 18th DoD/SERC meeting, and the notes below this charge represent the updated status as of the end of December (2015) with proposed FY 2016 activities shown, as well, for the multi-sentinel portion of the effort (topic 1 of 4).



RT-136: Part I, Section 4 of the Systems Engineering Research Center Topic Document (Feb 2015)

Appendix 12: Defense of Software Reconfigurable Radar Against Chronic Covert Cyber Attacks

Introduction

This study develops a strategy to defend a software reconfigurable radar (SRR) against chronic covert cyber attacks. The Georgia Tech Research Institute (GTRI) evaluated the SRR design based primarily on commercial off-the-shelf (COTS) components. In parallel this effort at the University of Virginia (UVa) developed a strategy to defend the SRR from an insidious type of attack.

The objectives of this study were to understand and characterize this type of attack, to evaluate the SRR structure within the context of the attack category, and to develop potential defenses that would likely exhibit a favorable balance between defender cost and attacker deterrence. A significant objective for the SRR was to attain application flexibility while managing cost, and that objective for the SRR imposed a companion objective to limit cost for any cyber defense mechanism while achieving defense flexibility appropriate for the varied missions of the SRR.

This UVa study initially developed and refined the definition of the chronic covert cyber attack. This study also observed the parallel work by GTRI to define the SRR architecture so that necessarily limited defender resources could be placed within the SRR efficiently. General defense tactics were then considered for the SRR architecture using the strategy of trusted sentinels to monitor SRR behavior and distinguish likely attacks from simple poor performance or malfunctions.

SRR Applications

Radars have traditionally been built for specific applications, and each new design is usually difficult and expensive. The SRR approach endeavors to define a common hardware architecture that can be tailored using software to satisfy a range of applications. Advantages of this SRR approach include reduced time to adapt the SRR to a new application at a lower cost than the traditional custom design. Application categories suggested for the SRR include maritime radar, aviation radar, terrestrial radar, and other applications such as signal intelligence and jamming.

The use of COTS components and an architecture common to these various applications can reduce cost and design effort, but this approach can also increase vulnerability to an insidious type of cyber attack. The commonality of architecture and the commercial sources for components offer potential attack vectors through malicious functionality embedded within the system itself either at the time of construction or any time thereafter. Thus the flexibility gained by the SRR concept comes at a cost of increased vulnerability.

Chronic Covert Cyber Attacks

This UVa study initially developed and refined the definition of the chronic covert cyber attack. This type of attack is particularly insidious because it can cause the defender to lose confidence in the system without recognizing that attacks have occurred. Each aspect of this type of attack establishes conditions that should be considered when establishing a defense strategy.

Cyber Attack

In the context of this study, a cyber attack is assumed to arise from within the system under attack. The mechanism of the attack already resides within the system under attack when the attack is triggered. The trigger may arrive from outside of the system or it may arise from some event or state within the system. The attack mechanism may have been placed in the system when it was first constructed, or it

may be placed at any time thereafter. For example, the attack mechanism may be placed in the system during routine maintenance of through malware arriving through communication channels. The imagination and resources of the adversary only the only real limits to the path to placement.

Covert Attack

In the context of this study, a covert attack deceives the defender so that the symptoms produced by the attack mechanism are not likely to be recognized as arising from an attack. System performance may be degraded or denied in a manner that appears to be just a benign malfunction. Examples of such an attack might include degraded signal to noise ratio (SNR), a dropped target track, changed operating mode, or even power generator failure. While many of these symptoms could appear in a manner suggestive of attack, more subtle presentations could degrade or deny service at critical times without being obviously caused by an attack.

Chronic Attack

In the context of this study, a chronic attack is one that can be repeated. This characteristic goes beyond the likelihood that the attack may be misinterpreted as a benign malfunction. The chronic characteristic also arises from both the non-destructive nature of the attack and the repetitive capability of the trigger.

Attack Mechanism Characteristics

The attack mechanism is introduced into the system in advance of its use, and this implies certain necessary characteristics.

- *Dormancy* – The attack mechanism must be hidden so that it will not be exposed through normal maintenance and test procedures. It must lie dormant and undetected.
- *Advantage* – The attack mechanism must achieve objectives that will be generally favorable to the attacker across a broad range of potential conditions. The attacker may not know at the time of mechanism installation what circumstances may exist at some unknown future time when the attack will be enabled.
- *Obfuscation* – The attack mechanism must produce symptoms that are simultaneously advantageous to the attacker and easily misinterpreted by the defender as poor performance of malfunction. This characteristic is necessary to support reuse.

These characteristics suggest that the attacker must have very good advance knowledge of the system under attack. This need for possibly detailed knowledge of the system is a disadvantage for the attacker that can be exploited by the defender.

Sentinel Defense Strategy

The Systems Engineering Research Center (SERC) has been engaged with the Department of Defense (DoD) in developing a novel cyber security concept for embedding security solutions into systems; this concept is referred to as System-Aware Cyber Security. These solutions provide greater assurance to the most critical system functions by providing an additional layer of defense that complements perimeter and network security solutions that serve to guard the entire system from penetration. System-Aware solutions are particularly effective at guarding against insider and supply chain attacks that circumvent perimeter security solutions. The broad objective of the System-Aware program can be thought of as reversing cyber security asymmetry from favoring our adversaries, to favoring the US; i.e., from favoring a small investment in straightforward cyber exploits to favoring small investments in System-Aware cyber security solutions for protecting critical system functions.

When trying to protect a Radar from a cyber-based attack, important questions arise when identifying priorities for potential threats, purposes, consequences and level of effort to achieve them: Which systems and functions, if compromised, can lead to significant disruption? What components or system configurations are inherently vulnerable to classes of cyber-attack? Where can these threats originate?

The selected elements of the system are then defended through the use of highly secure and trusted sentinels. The sentinel would be capable of monitoring the Radar subsystems, detecting when those subsystems have been compromised (i.e., when they have been altered through malicious activity), alert the appropriate authorities, and possibly taking actions to restore those subsystems to an uncompromised state.

Each sentinel is a computing platform separate from the system that it monitors and defends. The sentinel must be secure and trusted, and these characteristics can be challenging to attain. However, strict limitations on sentinel capabilities enable sentinel functionality on low-performance limited function platforms. These sentinels are kept simple and therefore are easier to validate and secure than the platforms that are being monitored.

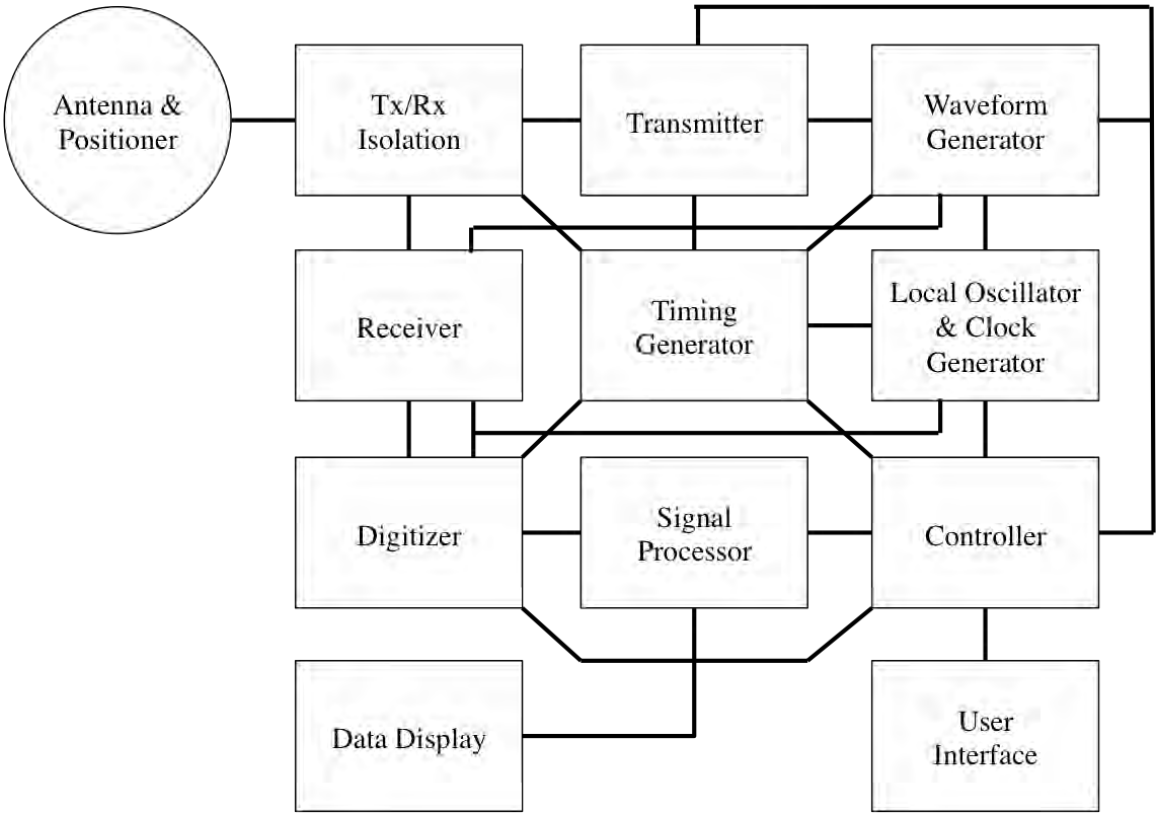
Sentinels are employed to monitor system functions to ensure that they are operating in a manner that is consistent with their design and configuration. If a sentinel detects an inconsistency, it should also determine whether the observed inconsistency likely arises from a cyber-attack. This sentinel defense strategy implements a system-aware solution.

There are several design issues and trade-offs that should be considered:

- Larger systems may use multiple sentinels distributed throughout a system to monitor subsystems and then coordinate on the detection and response to a cyber-attack. Alternatively, a single sentinel can monitor multiple subsystems and make determinations regarding detection and response.
- Generally speaking, inclusion of more functionality in sentinels incurs the expense of expanded and more complex software yielding potential reductions in the sentinel security.
- Sentinels can be secured through a variety of techniques because many (if not all) of a sentinel's monitoring and control functions require little software for implementation.
- Design concepts to support the development of groups of coordinating sentinels will need to be created to support security for functions that are executed across a system of systems. Sentinel designs must be predicated on a systematic approach for (1) identifying potential distributed attacks that require coordination for detection and response, (2) allocating functions to individual Sentinels, and (3) designing the needed secure coordination among Sentinels to detect and respond to such attacks.
- Operators will likely be engaged in making what could be critical response decisions in the face of a cyber-attack. The decisions will impact not only current operations, but also must account for intelligence gathering opportunities about the attack and attackers' objectives, the possibilities for device replacement instead of system restoration, the consequences of a wrong decision, especially in cases where the fail-over mode is inferior to the primary mode of operation.

The SRR Structure and Attack Susceptibility

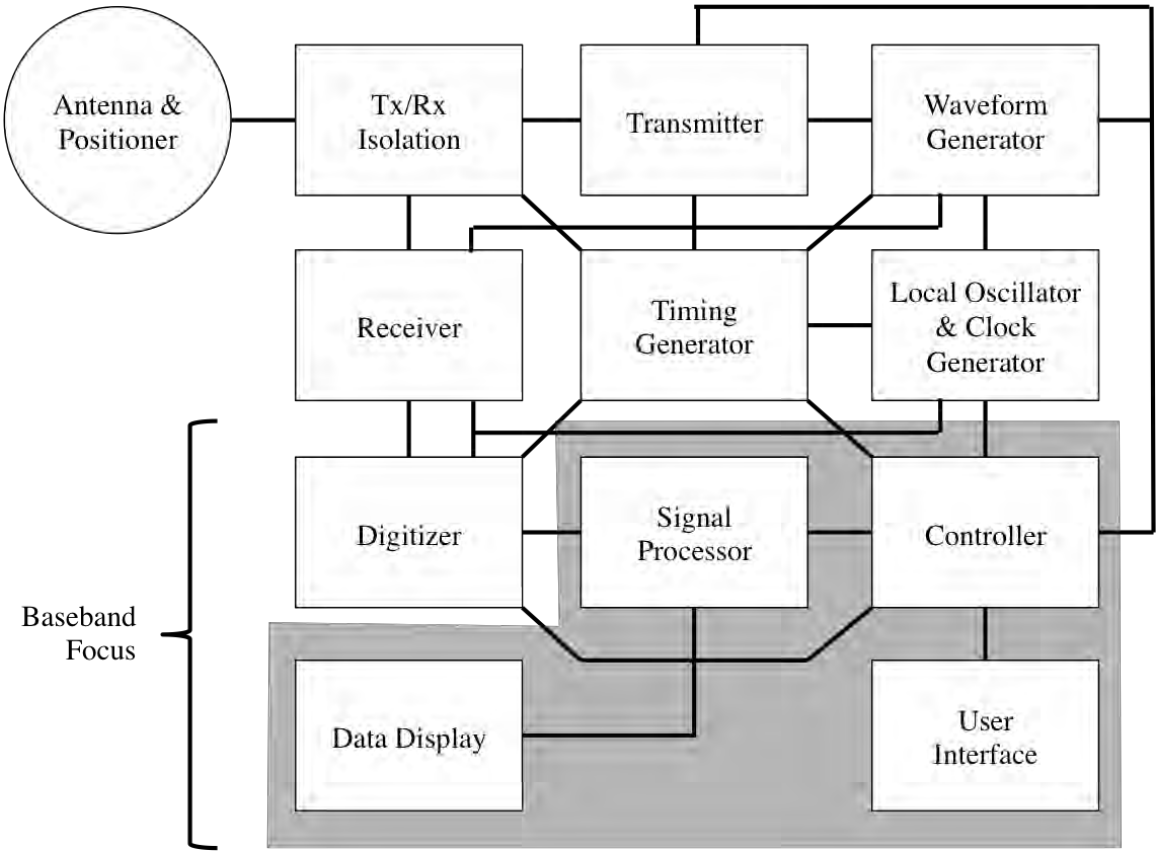
Appendix 12 Figure 1 shows a block diagram of an example radar system. Digital devices operating at baseband accomplish most of the sophisticated processing in radar. Transmission, reception, and other functions are performed at radio frequency (RF).



Appendix 12 Figure 1 Radar System Block Diagram

It would be possible to embed attack mechanisms in either the baseband or RF domains, but the baseband domain should be more attractive to the attacker. The size and complexity of the digital elements in the baseband domain provides more cover that can be used to hide the attack mechanism. Standard testing and maintenance would be more likely to expose attack mechanisms within the RF domain.

Appendix 12 Figure 2 identifies the radar blocks that are judged to be most susceptible to the type of attack considered in this study. Some baseband digital components may be found in other subsystems, but the greatest concentration of these components is in the four blocks identified in the figure.

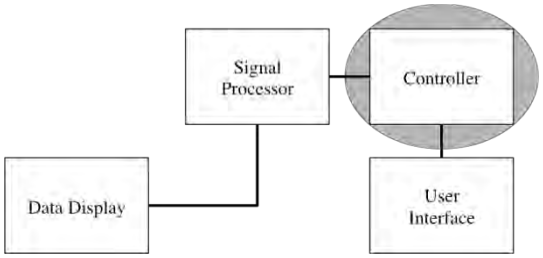


Appendix 12 Figure 2 SRR Susceptible Blocks

Another domain of attack susceptibility in a radar system is outside of the blocks shown in Appendix 12 Figure 2. The support infrastructure such as power and cooling could also be candidates for attack. Such an attack would deny access to the system services and could be implemented in a form that might be misinterpreted.

Example Sentinel Functions

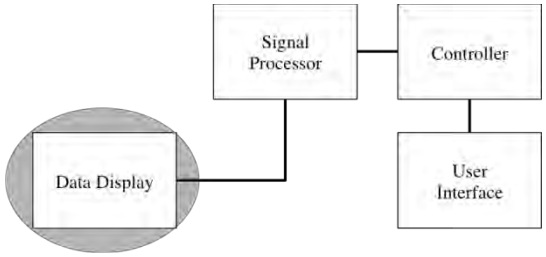
A Controller Sentinel



A sentinel could protect each of the functions represented by the baseband boxes. For example, a sentinel performing the following functions could monitor the radar controller:

- Controller inputs and outputs would be observed.
- The sentinel would perform "sanity checks" on control outputs.
- The sentinel would perform "sanity checks" on input combinations and controller state.

A Data Display Sentinel

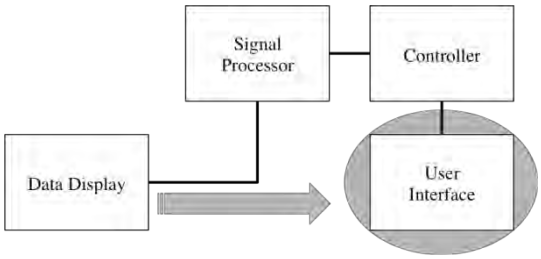


Alternately or in addition a sentinel could monitor the inputs and outputs of the data display. Note that this sentinel would not be checking target detection or track formation. Rather, this sentinel would be confirming that the image presented on the display is a reasonable

facsimile of the data presented to the display. Such a sentinel might perform the following functions:

- It would translate data presented to the display into a representation of the display format.
- It would capture an image of the display data presentation.
- It would compare the expected display with the observation.
- It would evaluate the display quality.

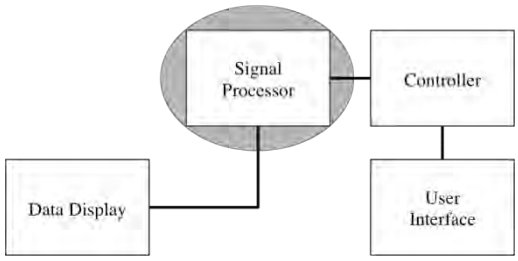
A User Interface Sentinel



In a similar manner, a sentinel could be used to monitor the user interface. This sentinel would observe information presented to the user and actions taken by the user. These observations would be used to determine whether user responses were reasonable for the presented information.

The user interface sentinel is not intended to assess any possible malicious action by the user. That type of attack would be outside of the types of attacks addressed by this study. Rather, this user interface sentinel would try to identify situations in which a compromised user interface might substitute an inappropriate user response for the response actually provided by the user.

A Signal Processor Sentinel



The final block not yet addressed is the signal processor. This block is typically very computationally demanding. It is responsible for finding targets and establishing tracks. Its size, complexity, and performance demands make it a good potential target for attack.

Several approaches have been considered for sentinel protection of the signal-processing block. Two example approaches use disparate redundancy of computation to allow the sentinel to determine whether the signal-processing block is performing as intended. Both of these approaches achieve disparate processing at limited cost.

One of these approaches attempts to perform generally equivalent but low resolution processing of the incoming data at full speed. The sentinel would provide much lower performance than the signal processor, but the resolution of the processing on the sentinel would be low enough to enable computation at speed. The results of the low resolution processing would be used to check whether the high-resolution results were in appropriate ranges.

The other of these approaches would perform the full high-resolution processing redundantly on the sentinel. The limited performance sentinel processor further burdened by its protection mechanisms would render it unable to complete the processing in real time. However, it would be able to complete full resolution computation of "snap shots" of the incoming data. These sampled intervals might be for one entire radar cycle, or for more or less than one cycle. Similar sampled processing could be performed across the results of multiple real time intervals to evaluate track formation.

Central Sentinel Functions

The primary objective for all sentinels is to determine whether the observed actions are reasonable for the circumstances. One approach to achieve this action involves redundant determination of results that can be compared. If redundant computations are performed on disparate platforms, and attacker would need to have detailed knowledge of more than one platform to produce incorrect results that still match.

Bounds for many values may be determined in advance, or normal operating bounds may be established and refined through operation.

Summary

This study has identified and refined a category of attack identified as a chronic covert cyber attack in the context of software reconfigurable radar. The general SRR architecture has been evaluated to identify most likely domains for this type of attack. The sentinel defense strategy has been described as an approach that should be appropriate for defending against this type of attack against an SRR. Example sentinel functionality has been suggested for use within the most vulnerable blocks of the SRR.